Tesis Doctoral

Diseño de un Algoritmo para la Revisión de Creencias entre Formas Conjuntivas

Tesis presentada para obtener el grado de Doctor en Ingeniería del Lenguaje y del Conocimiento

Presenta
Pedro Bello López

Director
Dr. Guillermo De Ita Luna

Junio 2021

Benemérita Universidad Autónoma de Puebla
Doctorado en Ingeniería del Lenguaje y del conocimiento

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Diseño de un Algorítmo para la Revisión de Creencias

Tesis presentada por Pedro Bello López
para obtener el grado de Doctor en Ingeniería del Lenguaje y del Conocimiento

2021

Ingeniería del Lenguaje y del Conocimiento

Agradecimientos

A mi familia por el apoyo durante el tiempo de desarrollo del doctorado.

A mis amigos y compañeros de la Facultad de Ciencias de la Computación por su buena vibra.

A mi asesor Dr. Guillermo De Ita Luna por su amistad, compañerismo y un excelente facilitador de conocimiento para llegar a culminar este proyecto de tesis.

A mis profesores del Doctorado en Ingeniería del Lenguaje y del Conocimiento por aportar en cada curso sus conocimientos para enriquecer el trabajo desarrollado.

Al fundador del programa de Doctorado en Ingeniería del Lenguaje y del Conocimiento, Dr. David Eduardo Pinto Avendaño por su amistad, sus conocimientos y por las facilidades brindadas durante toda la estancia en el doctorado.

A los miembros del comité académico por su ayuda en la revisión del proyecto de tesis desde sus inicios hasta la culminación del mismo.

Dra. Mireya Tovar Vidal (BUAP)

Dr. Jesús Ariel Carrasco Ochoa (INAOEP)

Dr. José Luis Carballido Carranza (BUAP)

Dr. Mauricio Javier Osorio Galindo (UDLAP)

Dr. Guillermo De Ita Luna (BUAP)

Al CONACYT por el apoyo económico brindado para el desarrollo de la tesis.

A Dios por darme salud.

Pedro Bello López

Índice general

1. Introducción **6**

1.1. Planteamiento del problema 11

1.2. Objetivos general y específicos 12

1.3. Justificación 13

1.4. Organización de la tesis 14

2. Marco teórico **16**

2.1. Lógica poposicional 17

2.1.1. Inferencia lógica 20

2.1.2. Formas Normales 22

2.1.3. Consecuencia lógica 24

2.2. Complejidad computacional del problema 25

2.2.1. Eficiencia 26

2.2.2. Clases de complejidad 28

2.2.3. Satisfactibilidad 32

2.3. Modelos de revisión de creencias 35

2.3.1. AGM 35

2.3.2. KM 38

3. Trabajo relacionado **39**

3.1. Revisión de creencias con lógica proposicional 40

3.2. Desarrollo de la teoría de revisión de creencias 42

3.3. Modelos y operadores de revisión de creencias 48

4. Metodología del proyecto de tesis **51**
4.1. Etapas del diseño de los algoritmos 52
4.2. Diseño del algoritmo de revisión de creencias 54
4.3. Algoritmo de revisión de creencias 59
4.3.1. Construcción del conjunto independiente de cláusulas 61
4.3.2. Propuesta algorítmica 62
4.3.3. Aplicación del algoritmo 66
4.3.4. Análisis de la complejidad del algoritmo 69
4.3.5. Postulados KM 71
4.4. Algoritmo de revisión de creencias con DFS 73
4.4.1. Aplicación del algoritmo DFS 79
4.4.2. Consistencia de la base de conocimiento 81

5. Diseño del Algoritmo de conteo de modelos **85**
5.1. Contracción de la base de conocimiento 90
5.1.1. Análisis del algoritmo 95

6. Conclusiones **97**
6.1. Aportación del trabajo de investigación 98
6.2. Trabajo a futuro 100

A. Algoritmos implementados **102**
A.1. Sistema de revisión de creencias 103
A.2. Sistema de conteo de modelos 105

Bibliografía **111**

Capítulo 1

Introducción

La revisión de creencias como lo cita Fermé [26], es una forma de representar el conocimiento (en una base de conocimiento) que un ser racional adquiere a través del tiempo, del estudio, por experiencia o simplemente de sus vivencias; cuando llega una nueva información la base de conocimiento se tiene que actualizar y si entra en contradicción con lo que ya sabemos será necesario restablecer la consistencia lógica de la base de conocimiento. Por ejemplo, a algunos de nosotros se nos enseñó en la educación básica que el Sistema Solar estaba formado por nueve planetas; sin embargo, a partir de 2006 [1] se acordó por la Unión Astronómica Internacional que Plutón no era como tal un planeta (fue reclasificado), por lo que ahora nuestro Sistema Solar consta de ocho planetas clásicos, esta nueva información hace que nuestra base de conocimiento tenga que ser modificada.

Por otra parte, nuestra base de conocimiento puede estar formada, además, por un conjunto de reglas o normas, incluso leyes que todo ser humano debe seguir o cumplir, y es en este sentido donde la inferencia lógica contribuye a determinar si la nueva información ya está presente en nuestra base de conocimiento o no. En [21] se considera que el razonamiento humano opera con base en reglas lógicas de inferencias, incluso en [93] propusieron que era posible modelar la actividad cerebral mediante la lógica. Así, si fuera posible modelar el comportamiento humano, los robots podrían emular nuestro comportamiento y facilitarnos la vida ejecutando tareas cotidianas de forma inteligente.

[1] https://www.muyinteresante.com.mx/preguntas-y-respuestas/pluton-no-se-considera-planeta/

El problema de revisión de creencias consiste en incorporar nuevas creencias a una base de conocimiento (en inglés *Knowledge Base* KB) ya establecida, con cambios mínimos a la KB para preservar las creencias originales y mantener la coherencia de la KB [71]. El paradigma más conocido para la revisión de creencias es el modelo AGM (por sus autores Alchourróm, Gärdenfords y Makinson) [17], desarrollaron un modelo para el cambio de creencias. En el modelo AGM la operación de revisión de creencias privilegia la nueva información con respecto a las creencias ya existentes en el conocimiento del agente. Otro punto importante es conocer la fiabilidad de la fuente de la nueva información, tal y como lo cita Liberatore, [78], lo cual permite reducir los tiempos de complejidad en tiempo en este proceso de revisión. Posteriormente, como se indica en [39], se unificaron los diferentes enfoques de revisión de creencias semánticas, y se reformularon los postulados AGM, llamándose ahora postulados KM (por sus autores H. Katsuno y A.O. Mendelzon).

Los científicos han buscado formas de cómo el cerebro requiere representar su base de conocimiento y han planteado que mediante los símbolos y conectores de un tipo de lógica, llamada proposicional, se puedan representar las creencias para la deducción de creencias. Se captura cada enunciado al que se le pude dar un valor lógico a través de las proposiciones, los que será denotados con letras del alfabeto ($p, q, r, s, t...$ etcétera) y representan oraciones (sujeto + predicado) vinculadas con la realidad que acontece. Por otro lado, los conectores que en lógica se les llama operadores lógicos permiten ligar y estructurar las oraciones. De esta forma cuando el cerebro reciba cada creencia (oración) se estructuran relaciones entre ellas. Así se tienen los operadores: $\neg$ (conocido como no) que cambian el sentido de la creencia original; si a un individuo le inculcan como norma: se debe terminar la comida del plato (proposición p), y en su realidad observa que no todos cumplen esa regla, entonces, no necesariamente se debe terminar la comida del plato. La oración anterior se expresaría como $\neg p$. Si además de la norma p, al individuo se le enseña la norma: comer con la boca cerrada (proposición q), en un determinado momento al individuo se le puede solicitar que al mismo tiempo se cumplan ambas normas. Al juntar estas dos normas da origen al operador ($\wedge$), es decir ($p \wedge q$), significa que ambas normas deben acontecer, por lo que el operador le permite al individuo asociar en su base de conocimiento que ambos sucesos requieren cumplirse para tener un comportamiento

deseado. Por otro lado, si al individuo se le inculcan las normas: puedes entregar al anfitrión un postre (proposición r), puedes entregar al anfitrión un vino (proposición s), el individuo es libre de decidir si lleva un postre o un vino o incluso ambos, por lo que en este caso estaría vinculado a través del operador $\vee$ para indicar que requiere cumplir al menos una de las normas o tiene la libertad de cumplir ambas. Para construir la base de conocimiento estos operadores pueden interactuar con estructuras llamadas cláusulas, donde el requisito para formarlas es que las oraciones se asocien con el operador $\vee$, indicando que se tienen una serie de alternativas pero que al menos se cumpla una, por ejemplo, $(p \vee q)$ y luego para generar la base de conocimiento se conectan mediante el operador $\wedge$, por ejemplo, $(p \vee q) \wedge (r \vee s)$.

Así, tener dos o más cláusulas conectadas significa que la base de conocimiento es consistente si por cada cláusula en su estructura interna se cumple al menos una alternativa y cada cláusula debe cumplirse para el buen funcionamiento de la base. Cuando se tienen conjuntos de cláusulas conectadas mediante el operador $\wedge$, a esa nueva estructura en lógica se le conoce como Forma Normal Conjuntiva (FNC). Una fórmula está en Forma Normal Conjuntiva si es una conjunción de disyunciones de literales, es decir, es de la forma $(x_{1,1} \vee \cdots \vee x_{1,n1}) \wedge \cdots \wedge (x_{m,1} \vee \cdots \vee x_{m,n_n})$.

La inferencia lógica significa deducir algo, sacar una conclusión o conducir un nuevo resultado a partir de algo conocido, mediante un proceso mental que crea consecuencias con sentido a la situación previa que ocurra. Por ejemplo, cuando a partir de una causa se determina un efecto, a esta relación entre ambos elementos, se le identifica como una implicación (inferencia) lógica. Una vez que se tenga un FNC se puede aplicar el operador de inferencia, que denotaremos con el símbolo $\models$. Cuando agregamos una cláusula ϕ (considerada nueva información) con el fin de verificar si se puede inferir o no esa cláusula a partir de la base de conocimiento; esta operación se denota como $K \models \phi$.

En el área de Inteligencia Artificial, el razonamiento en los sistemas inteligentes es el enfoque de sistemas basado en el conocimiento, el cual consiste en mantener el conocimiento en algún lenguaje de representación con connotación bien definida [36]. El conocimiento de un agente se cierra bajo la consecuencia lógica $\mathrm{Cn}(K)$, es decir, un agente creerá todas las consecuencias lógicas de sus creencias, estas sentencias se almacenan en una base de conocimiento K provista de un mecanismo

de razonamiento deductivo [31].

La implicación proposicional [20] es una tarea importante en problemas tales como la estimación del grado de creencia y actualización de las creencias, en procedimientos con aplicaciones de la Inteligencia Artificial, en la planificación y diseño de sistemas multi-agente, en el diagnóstico lógico, el razonamiento aproximado y en el conteo del número de soluciones para instancias de satisfactibilidad [66], entre otras aplicaciones.

En general, el problema de la implicación lógica es un reto para el razonamiento automático, ya que se conoce que es un problema perteneciente a la clase Co-NP completo, como lo muestran [89] y [77]. Se han realizado grandes esfuerzos para reducir los costos computacionales asociados con el razonamiento automático, mediante la compilación de la base de conocimiento, y usando fórmulas restringidas para las nuevas entradas, todo esto con el fin de reducir la tarea de verificar inferencia. En este sentido, las cláusulas de Horn se han utilizado como el estándar de fórmulas de la lógica para tareas de inferencia, debido a que existen procedimientos de inferencia eficiente para esta clase de fórmulas.

Otros modelos que se identifican por el nombre de sus autores, Dalal, Satoh, Winslett, Borguida y Forbus son citados por [80] como modelos a considerar en la revisión de creencias. El operador descrito por [55] sugiere la revisión en base a la distancia mínima de Hamming entre interpretaciones y bases de conocimiento. En la práctica, esta propuesta consiste en el cálculo del conjunto de modelos, lo que es muy costoso computacionalmente. Uno de los inconvenientes del enfoque de Dalal es limitado en el caso de bases de conocimiento consistente. Por lo tanto en [97] se propone un nuevo método de cálculo Dalal para que en el proceso de revisión se evite el cálculo de modelos de bases de creencias. Sin embargo éste sólo funciona para una forma normal disyuntiva mientras que en [52] se propone una nueva distancia mínima, en lugar de la distancia de Hamming, para adoptar una creencia como un hecho. Otro de los modelos relevantes se debe a Darwiche [4] quien propuso la revisión iterada de creencias, dicha propuesta establece una representación basada en los supuestos del modelo manteniendo la consistencia en el proceso de revisión de creencias, este modelo se le conoce como DP [3].

La propuesta descrita por [49] es similar a la de Dalal, con la diferencia de que

la distancia entre los dos modelos se define como el conjunto de literales diferentes entre ambos modelos. En el caso de Winslett, la propuesta se basa en una posible comparación entre todos los máximos sistemas coherentes. Recientemente, la revisión de creencias ha despertado interés en el marco de la lógica simbólica, incluyendo cláusulas de Horn. Muchos de los autores que han generado operadores para revisión de creencias convergen en puntos de vista sintáctico y semánticos, como lo muestran las propuestas de [80], [14] y [49].

Es importante mencionar que el problema de revisión de creencias y en particular, usando lógica proposicional como lo cita [70] es considerado un problema difícil que pertenece a la clase Co-NP. Así en [80] y [46] se describe de forma simple que de acuerdo a la teoría de la complejidad denotamos la clase P como el conjunto de problemas cuya solución puede ser resuelto en tiempo polinomial por una máquina de Turing determinista, mientras que NP denota la clase de problemas que pueden ser resueltos en tiempo polinomial por una máquina de Turing no determinista (en la sección 2.2 se amplian estos conceptos).

Hay una importante clase de problemas para los que no se han descubierto algoritmos polinomiales. Para los problemas de esta clase (los llamados problemas NP-completos) sólo se conocen algoritmos que, en el caso peor, necesitan un número exponencial de pasos. Esta clase incluye miles de problemas prácticos importantes que surgen, por ejemplo, al trazar rutas de transporte o redes de comunicación, asignar máquinas u otros recursos en procesos industriales, confeccionar horarios de escuelas, líneas aéreas, cargar un camión o un barco, etc. En la subclase NPC de los problemas NP-completos se consideran los problemas más dificiles y característicos de la clase NP. Los problemas tales que sus complementos están en NP se denomina la clase CO-NP. Un problema de CO-NP es CO-NP-completo si todos los problemas de CO-NP se reducen polinomialmente a él. Se puede demostrar que si un problema es NPcompleto, entonces su complemento es CO-NP-completo. Si CO-NP es la clase de los problemas complemento de los de NP, se cumple que $P \subseteq NP \cap CO\text{-}NP$ [90].

1.1. Planteamiento del problema

La revisión de creencias consiste en la dinámica de cómo deben cambiar los estados epistémicos de un agente como resultado de recibir nueva información para almacenarse en su base de conocimiento, el caso más simple (expansión) surge por el aprendizaje de algo nuevo, pero en ocasiones la nueva información contradice creencias previamente aceptadas y entonces es necesario eliminar (contracción) viejas creencias.

En los últimos años se han realizado diferentes propuestas para modelar el problema de revisión de creencias a través de proponer operadores para la revisión, actualización y eliminación de creencias. Sin embargo, los intentos se centran en el proceso en general de revisión de creencias, y pocas veces consideran la complejidad computacional al actualizar la base de conocimiento.

Un marco ampliamente aceptado para el razonamiento en sistemas inteligentes es el enfoque de sistemas basado en el conocimiento como lo describe [15] en su tesis doctoral. La idea general es mantener el conocimiento en algún lenguaje de representación con una connotación bien definida. En la Figura 1.1 se describe este proceso, donde las sentencias se almacenan en una base de conocimiento provista de mecanismos de razonamiento, para lo cual se utiliza generalmente razonamiento proposicional deductivo. Cuando la nueva información ϕ no se infiere de K, entonces se genera nueva información denotada como S que puede ser agregada a la base K (indicado como $K + S$) de tal forma que ahora se cumpla que $K + S$ infiere a ϕ. Este proceso será realizado por el operador de revisión de creencias.

El problema a tratar en este trabajo de tesis es proponer un algoritmo para la revisión y actualización de una base de conocimiento usando lógica proposicional. Una base de conocimiento almacena conocimiento en una forma legible para la computadora, usualmente con el fin de obtener razonamiento deductivo automático aplicado a ellas, las cuales contienen una serie de datos en forma de reglas que describen el conocimiento de manera lógicamente consistente usando operadores lógicos.

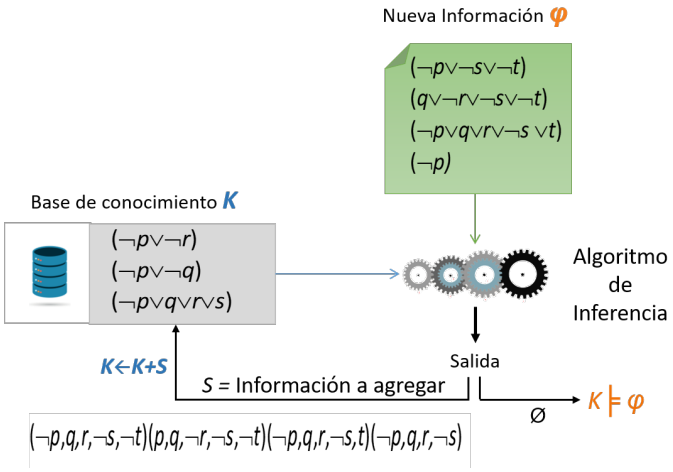


Figura 1.1: Modelo base de inferencia propocicional

1.2. Objetivos general y específicos

Objetivo general. Diseñar un operador de revisión de creencias entre formas normales conjuntivas utilizando métodos de deducción automática con el fin de contribuir a delimitar la frontera entre problemas tratables e intratables.

Objetivos específicos.

- Diseñar la propuesta del operador de revisión de creencias considerando los casos de prueba para que se cumplan los postulados KM y AGM.
- Verificar la correctez y/o completitud de la propuesta del operador de revisión desarrollado para formalizar la propuesta.
- Realizar el cálculo de la complejidad en tiempo del algoritmo desarrollado para compararse con otros operadores del estado del arte.
- Generar un conjunto de instancias de prueba en lógica proposicional para proponer posibles soluciones al problema de revisión de creencias.

1.3. Justificación

Las teorías de revisión de creencias han experimentado un fuerte desarrollo desde 1973 hasta los años recientes, y se espera que dicho desarrollo continúe.

En la Figura 1.2 se muestra el desarrollo de artículos (fuente: scopus.com) con el tema revisión de creencias desde los inicios de esta teoría (1985) hasta la fecha actual, durante este periodo de tiempo se ha mantenido un crecimiento en la publicación de material referente al tema, lo que puede sugerir una importancia relevante en el desarrollo de la ciencia.

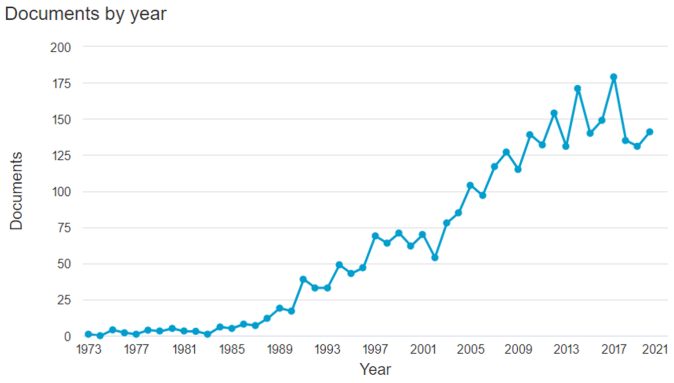


Figura 1.2: Producción de artículos con el tema Revisión de Creencias (fuente: scopus.com).

De los documentos publicados más del 55 % corresponden a artículos y más del 30 % son publicaciones de conferencias, existen otro tipo de publicaciones como "*reviews*", libros, capítulos de libro, etc., como se indica en la Figura 1.3.

- Entre las características de los trabajos en revisión de creencias, se puede comentar que: las teorías de revisión de creencias constituyen un enfoque relativamente novedoso en el razonamiento de agentes, por lo que explorar esta área del conocimiento con operadores de revisión de creencias permite modelar problemas prácticos.

- El establecer métodos del cálculo de la complejidad en tiempo de los operadores de revisión de creencias aporta un marco que genera conocimiento en ciencias computacionales.

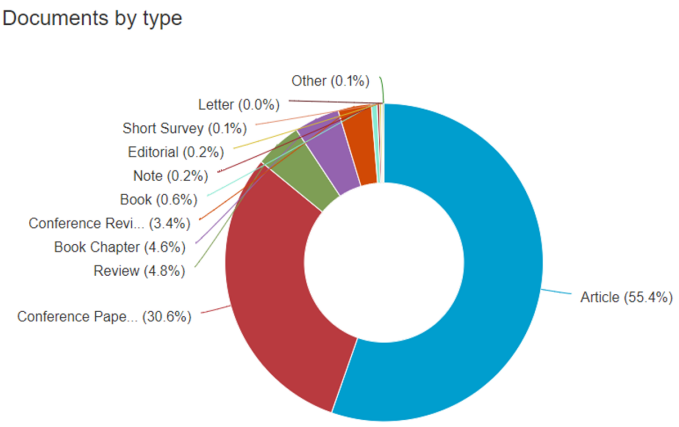


Figura 1.3: Producción por tipo de artículo con el tema Revisión de Creencias (fuente: scopus.com).

- Aplicar teorías de lógica proposicional, lógica de predicados, lógica deóntica entre otras posibilidades amplia el marco de estudio de la teoría de revisión de creencias.

- Las teorías de revisión de creencias, resultan atractivas para todos aquellos interesados en razonamientos no monotónicos (cualquier sistema de razonamiento que utilice reglas ampliadas de inferencia tiene necesariamente la propiedad de ser no monotónico). En general ϕ se infiere no monotónicamente de K, dado una KB, si y sólo si ϕ pertenece al conjunto K revisado con ϕ.

- Los modelos de revisión de creencias pueden servir además para comprender y evaluar cambios en sistemas legales como en la lógica deóntica.

1.4. Organización de la tesis

A continuación se describe la organización de la tesis.

En el capítulo 2 se describen los conceptos básicos requeridos para plantear el trabajo de investigación en el diseño de un algoritmo para la revisión de creencias, se incluyen los temas de lógica proposicional, inferencia proposicional, los principales

postulados de la teoría de revisión y la complejidad computacional relacionado al problema de revisión de creencias.

En el capítulo 3 se presenta el trabajo relacionado a través de la historia del desarrollo de la teoría de revisión de creencias desde sus origenes en 1985 (y anteriores) hasta el año actual. Concluyendo este capítulo con un tabla resumen de los principales modelos y los principales operadores de revisión de creencias desarrollados.

En el capítulo 4 se presenta el diseño del algoritmo de revisión de creencias. Se describe cada una de las fases en el diseño de las propuesta algorítmica y se muestran los postulados KM. También se plantea de manera general la propuesta del orden de complejidad en tiempo del algoritmo.

En el capítulo 5 se presenta el diseño del algoritmo para verificar la consistencia de la base de conocimiento a través de una propuesta de conteo de modelos de una fórmula en FNC.

En el capítulo 6 se presentan las conclusiones del trabajo desarrollado, las principales aportaciones y algunas ideas del trabajo a futuro.

En el anexo A se presentan dos aplicaciones computacionales para ejemplificar la aplicación del algoritmo de revisión de creencias y de conteo de modelos.

Capítulo 2

Marco teórico

En el presente capítulo se describen los conceptos teóricos necesarios para el desarrollo del trabajo de tesis, en el diagrama de la Figura 2.1 se enmarca la revisión de creencias con los diferentes conceptos asociados al problema y las ramas indican el nivel de aplicación del proyecto de tesis.

Figura 2.1: Teoria básica a considerar en la revisión de creencias

2.1. Lógica poposicional

Un tema central en el desarrollo de éste proyecto de tesis es la lógica proposicional. La lógica proposicional como se indica en [22] se utiliza para analizar razonamientos formalmente válidos partiendo de proposiciones y operadores lógicos, con el objetivo de construir fórmulas formadas por operaciones lógicas entre variables proposicionales. La lógica proposicional se basa en proposiciones, donde una proposición es una oración que afirma o niega algo y por lo tanto puede ser verdadera o falsa. La lógica proposicional es un sistema lógico que busca formalizar la noción de la proposición como un conjunto de proposiciones [33].

La lógica proposicional, también conocida como lógica de enunciados, es un sistema formal cuyos elementos representan proposiciones o enunciados [22]. La revisión de creencias ha sido ampliamente estudiada en el marco de la lógica proposicional, pero recientemente la revisión dentro de fragmentos de la lógica proposicional ha ganado atención [62].

Uno de los objetivos de la lógica proposicional o de predicados en las ciencias de la computación es el desarrollo de lenguajes para modelar tanto las situaciones, los problemas y los algoritmos que los resuelven, y de tal forma, que se pueda razonar de manera formal sobre los modelos propuestos. Razonar sobre las situaciones significa la construcción de argumentos acerca de éstos, de forma que los argumentos sean válidos y puedan ser defendidos de forma rigurosa, o ejecutados de forma automática en una computadora.

La mayoría de las veces, cuando se usa una lógica para el diseño, especificación y verificación de sistemas computacionales se plantea la relación: $K \models \phi$, que significa determinar si la fórmula ϕ se implica lógicamente de K, donde K es la situación o el conocimiento actual sobre una situación, y ϕ es una fórmula lógica que codifica la información que deseamos saber si es verdadera bajo la suposición K, o que ϕ es satisfactible en K.

Desde el punto de vista computacional [33], se pueden diseñar e implementar algoritmos para calcular el operador de implicación lógica o de razonamiento ($\models$). La implicación proposicional de acuerdo a Cresto [3] es una tarea importante en problemas tales como la estimación del grado de creencia y actualización de las creencias, en el trabajo con procedimientos y en las aplicaciones de la Inteligencia

Artificial. Por ejemplo, cuando se trabaja en la planificación y diseño de sistemas multiagente, en el diagnóstico lógico, el razonamiento aproximado y en el conteo del número de soluciones para instancia de satisfactibilidad, como se indica en [66], entre otras aplicaciones.

En lógica proposicional, el *vocabulario* es un conjunto de símbolos de letras proposicionales P cuyos elementos normalmente se escriben como $p, q, r, \ldots$. Una fórmula de la lógica proposicional sobre P se define como:

- Todo símbolo proposicional de P es una fórmula.
- Si F y G son fórmulas, entonces $(F \wedge G)$ y $(F \vee G)$ son fórmulas.
- Si F es una fórmula, entonces $\neg F$ es una fórmula.

Una *Interpretación* (también llamada asignación) s sobre el vocabulario P es una función $s : P \to \{0, 1\}$, es decir, s es una función que, para cada símbolo proposicional, nos dice si es $verdadero$ o $falso$.

Sean s una interpretación y F una fórmula, ambas sobre el vocabulario P. La evaluación en s de F, denotada como $v_s(F)$, es una función que por cada fórmula da un valor de $verdadero$ o $falso$. Si $v_s(F) = verdadero$, se dice que s satisface a F, o que s es un modelo de F, es decir, una *interpretación* s es un modelo para una fórmula F si $v_s(F) = verdadero$.
Tautología: Para toda interpretación $s, v_s(F) = Verdadero$.
Contradicción: Para toda interpretación $s, v_s(F) = Falso$.
Satisfactible: Existe una interpretación $s, v_s(F) = Verdadero$.
Insatisfactible: No existe una interpretación $s, v_s(F) = Verdadero$.

En la lógica proposicional a $\wedge$, $\vee$ y $\neg$ se les llama *conectivas lógicas*. A menudo tambien se utilizan los conectivos $\rightarrow$ o $\leftrightarrow$, los cuales pueden ser expresados como abreviaturas de $(\neg F \vee G)$ y $((\neg F \vee G) \wedge (\neg G \vee F))$ respectivamente.

Formalmente para cada fórmula F existe un único *árbol* que representa la construcción de F según la sintaxis de la lógica proposicional. Por ejemplo la fórmula $((p \wedge q) \vee \neg(r \wedge \neg q))$ se puede reprentar como se indica en la Figura 2.2.

Definición 2.1 *Sea $X = \{x_1, \ldots, x_n\}$ un conjunto de n variables booleanas. Una literal denotada como lit, es una variable x_i o la negación de la variable $\neg x_i$, usualmente, cada $x \in X$, $x^0 = \neg x$ y $x^1 = x$.*

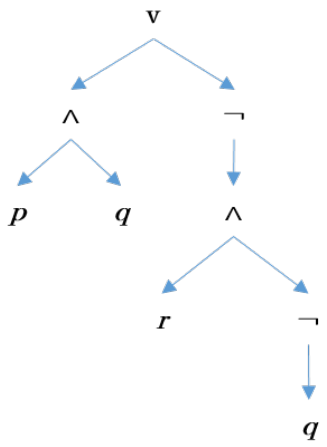


Figura 2.2: Representación por árbol semántico

Definición 2.2 *Una cláusula es una disyunción de diferentes literales. Para $k \in N$, $k-clausula$ es una cláusula con exactamente k literales, en general una cláusula es un conjunto de literales.*

Definición 2.3 *Una frase es una conjunción de literales. Una $k-frase$ es una frase con exactamente k literales. Una variable $x \in X$ aparece en una cláusula C (o frase) si x o $\neg x$ es un elemento de C.*

Definición 2.4 *Una fórmula F está en Forma Normal Conjuntiva (FNC) si es una conjunción finita de disyunciones de literales, es decir, es de la forma: $(L_{1,1} \vee \ldots \vee L_{1,n1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{n,n_m})$.*

Definición 2.5 *Una fórmula F está en Forma Normal Disyuntiva (FND) si es una disyunción finita de conjunciones de literales, es decir, es de la forma: $(L_{1,1} \wedge \ldots \wedge L_{1,n1}) \vee \ldots \vee (L_{m,1} \wedge \ldots \wedge L_{n,n_m})$.*

Una FNC F se satisface por una asignación s si cada cláusula de F se satisface por s. Un modelo de F es una asignación $v_s(F)$ que satisface F.

Dos cláusulas C_i y C_j son complementarias si C_i contiene una literal x, mientras que en Cj aparece la literal $(1-x)$.

Definición 2.6 *Dadas dos cláusulas C_i and C_j, si tienen al menos una literal complementaria, entonces son llamadas cláusulas independientes, de otra forma, se dice que ambas cláusulas son dependientes.*

Definición 2.7 *Sea $K = \{C_1, C_2, \ldots, C_m\}$ una FNC K es llamada independiente si para cualquier par de cláusulas $C_i, C_j \in K$, $i \neq j$, se encuentra la propiedad de independencia.*

Sea K una FNC. El conjunto de asignaciones que forman $Fals(C_i)$ se puede representar mediante una cadena A_i del conjunto {0, 1, *}. Si $C_i = \{x_{i_1} \vee \ldots \vee x_{i_k}\} \in K$, entonces el valor para cada posición de $i_1 - th$ a $i_k - th$ de la cadena A_i, tiene que falsificar las literales de C_i. Si $x_{i_j} \in C_i$ entonces el i_j - ésimo elemento de A_i se establece en 0. Si $\neg x_{i_j} \in C_i$ entonces el i_j - ésimo elemento de A_i se establece en 1. Las variables que no aparecen en C_i se representan por el símbolo *, lo que significa que podrían tomar cualquier valor lógico {0, 1}.

Definición 2.8 *Dadas dos cadenas falsificantes A y B cada una de longitud n, si hay una $i \in \{1, \ldots, n\}$ tal que $A[i] = x$, $B[i] = 1 - x$, $x \in \{0, 1\}$ se dice que tienen la propiedad de independencia, de otra manera se dice que ambas cadenas son dependientes.*

Definición 2.9 *Dado un par de cláusulas C_1 y C_2, si $lit(C_1) \subseteq lit(C_2)$ entonces se dice que C_2 es subsumida por C_1 y en este caso SAT(C_2) es (subconjunto o igual) a Sat(C_1).*

Dada una fórmula F, $S(F)$ es el conjunto de todos los posibles mapeos definidos en $v_s(F)$. Si $n = |v_s(F)|$ entonces $|S(F)| = 2^n$. $s \models F$ significa que la asignación s es un modelo de F (s satisface F). $s \not\models F$ denota que s es una asignación falsificante de F. Denotamos por $Sat(F)$ al conjunto de asignaciones en $S(F)$ para el cual son modelos de F, $Fals(F)$ denota el conjunto de asignaciones en $S(F)$ que falsifica F. De manera similar, $\#SAT(F)$ y $\#Fals(F)$ denotan el número de modelos y de falsificaciones de F respectivamente.

2.1.1. Inferencia lógica

La inferencia lógica y en particular la proposicional permite determinar si, dado un conjunto de reglas es posible inferir posibles conflictos. En [41] y [91] describen la manera en que sucede esto en la lógica deóntica ("la lógica de lo que debe ser", de

lo obligatorio y lo prohibido). Deon viene del griego que significa "lo que debe ser". El siguiente ejemplo aclarará el concepto:

De las conductas indecorosas en la mesa de mi señor (texto anónimo atribuido a Leonardo Da Vinci, quien trabajó para los Médici, 1600), (Anfrix, 2006). Estos son los hábitos indecorosos que invitados a la mesa de mi señor no deben cultivar:

- Ningún invitado ha de sentarse sobre la mesa, ni de espaldas a la mesa, ni sobre el regazo de cualquier otro invitado.
- Tampoco ha de subir los pies sobre la mesa.
- Tampoco ha de sentarse bajo la mesa en ningún momento.
- No ha de limpiar su armadura sobre la mesa.
- No ha de tomar comida de la mesa y ponerla en su bolso o faltriquera para después comerla.
- No ha de hacer figuras modeladas ni prender fuegos ni adiestrarse en hacer ruidos en la mesa (a menos que mi señor se lo pida).
- No ha de tocar el laúd o cualquier otro instrumento que pueda ir en perjuicio de su vecino de mesa (a menos que mi señor se lo pida).
- No ha de cantar, ni hacer discursos, ni vociferar improperios ni tampoco proponer acertijos obscenos si está sentado frente a una dama.
- No ha de conspirar en la mesa (a menos que lo haga con mi señor).
- Tampoco ha de prender fuego a su compañero mientras permanezca en la mesa.
- No ha de golpear a los sirvientes (a menos que sea en defensa propia).

Para mostrar la importancia de la inferencia lógica y la revisión de creencias modelamos algunos de los hábitos indecorosos en la mesa de mi señor:

- Ningún invitado ha de sentarse sobre la mesa, ni sobre el invitado.
- Tampoco ha de subir los pies sobre la mesa.

- No ha de limpiar su armadura sobre la mesa.
- No ha de prender fuego sobre la mesa.
- Tampoco ha de prender fuego a su compañero mientras permanezca en la mesa.

Cada una de las sentencias generadas se debe transformar a un lenguaje matemático como, por ejemplo, considerando los siguientes enunciados: p = sobre la mesa, q = sentarse, r = subir los pies, s = prender fuego, t = sobre invitado, u = limpiar armadura. Una vez representada cada oración (proposición) se conectan como cláusulas: $(\neg q \vee \neg p) \wedge (\neg q \vee \neg t)$, $(r \vee p)$, $(\neg u \vee \neg p)$, $(\neg p \vee \neg s)$, $(\neg p \vee \neg t \vee \neg s)$

En general, el problema de la implicación lógica es un reto para el razonamiento automático, ya que se conoce que es un problema en la clase Co-NP completo, incluso en el caso proposicional, como lo muestran Khardon et al. [5] y Liberatore [6]. En el desarrollo de algoritmos que resuelvan de forma automática el problema de implicación lógica, se utilizan algoritmos originalmente diseñados para resolver un problema relacionado: el problema de Satisfactibilidad (SAT).

2.1.2. Formas Normales

F y G son equivalentes si $v(F) = v(G)$ para toda valuación v, se representa, $F \equiv G$. Algunas equivalencias notables son:

- Idempotencia: $F \vee F \equiv F$ y $F \wedge F \equiv F$.
- Conmutativa: $F \vee G \equiv G \vee F$ y $F \wedge G \equiv G \wedge F$.
- Asociativa: $F \vee (G \vee H) \equiv (F \vee G) \vee H$ y $F \wedge (G \wedge H) \equiv (F \wedge G) \wedge H$.
- Absorción: $F \wedge (F \vee G) \equiv F$ y $F \vee (F \wedge G) \equiv F$.
- Distributiva: $F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$ y $F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H)$.
- Doble negación: $\neg\neg F \equiv F$.
- De Morgan: $\neg(F \wedge G) \equiv \neg F \vee \neg G$ y $\neg(F \vee G) \equiv \neg F \wedge \neg G$.

Para transformar una fórmula proposicional F a su forma normal conjuntiva, se siguen los siguientes pasos:

1. Eliminar los bicondicionales usando equivalencia:
   $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Eliminar los condicionales usando equivalencia:
   $A \rightarrow B \equiv \neg A \vee B$

3. Incorporar las negaciones usando las equivalencias:
   $\neg(A \wedge B) \equiv \neg A \vee \neg B$
   $\neg(A \vee B) \equiv \neg A \wedge \neg B$
   $\neg\neg A \equiv A$

4. Incorporar las disyunciones usando las equivalencias:
   $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
   $(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C)$

Para transformar una fórmula proposicional F a su forma normal disyuntiva, se siguen los siguientes pasos:

1. Eliminar los bicondicionales usando equivalencia:
   $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Eliminar los condicionales usando equivalencia:
   $A \rightarrow B \equiv \neg A \vee B$

3. Incorporar las negaciones usando las equivalencias:
   $\neg(A \wedge B) \equiv \neg A \vee \neg B$
   $\neg(A \vee B) \equiv \neg A \wedge \neg B$
   $\neg\neg A \equiv A$

4. Incorporar las disyunciones usando las equivalencias:
   $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$
   $(A \vee B) \wedge C \equiv (A \wedge B) \vee (B \wedge C)$

Teorema 2.1 *Para cualquier fórmula cerrada se puede encontrar una fórmula equivalente en Forma Normal Conjuntiva (disyuntiva).*

Prueba 2.1 *Cualquier fórmula se puede transformar en una fórmula en Forma Normal (disyuntiva) equivalente mediante las leyes de De Morgan y las propiedades distributivas.*

2.1.3. Consecuencia lógica

Definición 2.10 *Una fórmula F es consecuencia lógica (o consecuencia semántica) de un conjunto de fórmulas U, $U \models F$, si todo modelo de U es modelo de F, es decir, para toda valuación, v, $v \models U \Rightarrow v \models F$.*

Sean F, G y H fórmulas proposicionales, las siguientes son algunas propiedades de la consecuencia semántica.

- $F \models F$, toda fórmula es consecuencia semántica de sí misma.
- $F \models G$ y $G \models H$ entonces $F \models H$, transitividad.
- Si $G \in F$ entonces $F \models G$
- Si $F \models G$ y $F \subseteq Z$ entonces $Z \models G$, si de un conjunto de hipótesis se sigue una conclusión, agregando hipótesis al conjunto, la conclusión sigue siendo consecuencia semántica.

La consecuencia lógica se basa en los valores de verdad de las proposiciones atómicas que ocurren en las premisas y la conclusión; así como la forma en que los operadores lógicos manipulan estos valores de verdad.

Desde el punto de vista del razonamiento, tenemos que, un razonamiento es válido si la conclusión es consecuencia lógica de las premisas
$\Gamma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} : \Gamma \models C$
sii $\sigma_1, \sigma_2, \ldots, \sigma_n \Rightarrow C$
sii $\models \sigma_1, \sigma_2, \ldots, \sigma_n \rightarrow C$

La estructura de la forma $\{\sigma_1, \sigma_2, \ldots, \sigma_n\} \models C$ se denomina razonamiento, donde $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ es el conjunto de premisas y C la conclusión.

Decidibilidad

Un sistema deductivo completo puede demostrar cualquier conclusión lógicamente implicada por las premisas, lo que hemos denotado con $K \models \phi$. ¿Pero qué ocurre si $K \not\models \phi$?

En la teoría de la computación se define un problema de decisión como aquél que tiene dos respuestas posibles: "sí" o "no". Un problema de decisión es decidible si

existe un algoritmo (es decir, un procedimiento que siempre termina) que lo resuelve. En el marco de los sistemas deductivos es muy relevante también la cuestión de la decidibilidad, que se formula de la siguiente manera. Dados K y ϕ, ¿existe algún algoritmo que responda "sí" en el caso de que $K \models \phi$ y que responda "no" en el caso de que $K \not\models \phi$? claramente en la lógica proposicional esta cuestión tiene una respuesta positiva, por ejemplo utilizando algoritmos basados en las tablas de verdad [22].

2.2. Complejidad computacional del problema

Un **algoritmo** es cualquier procedimiento computacional bien definido que toma algún valor, o conjunto de valores, como **entrada** y produce algún valor, o conjunto de valores, como **salida**. Un algoritmo es por lo tanto una secuencia de pasos computacionales que transforman la entrada en la salida [37].

También podemos ver un algoritmo como una herramienta para resolver un problema computacional bien definido. La declaración del problema específica en términos generales la relación de entrada/salida. El algoritmo describe un procedimiento computacional específico para lograr esa relación de entrada/salida.

Se dice que un algoritmo es **correcto** si, para cada instancia de entrada, se detiene con la salida correcta. Decimos que un algoritmo correcto resuelve el problema computacional determinado. Un algoritmo **incorrecto** podría no detenerse en absoluto en algunas instancias de entrada, o podría detenerse con una respuesta incorrecta. Contrariamente a lo que se puede esperar, algoritmos incorrectos a veces pueden ser útiles, si podemos controlar su tasa de error.

Suele decirse también que un algoritmo es bueno si para ejecutarlo se necesita poco tiempo y se requiere poca memoria. En el análisis de algoritmos siempre se escogerá un tipo de operación particular que ocurra en el algoritmo, y se debe realizar un análisis matemático a fin de determinar el número de operaciones necesarias para completar el algoritmo.

2.2.1. Eficiencia

Cuando tenemos que resolver un problema, es posible que estén disponibles varios algoritmos adecuados, para lo cual debemos elegir entre varias posibilidades. El efoque *empírico (o a posteriori)* para seleccionar un algoritmo consiste en programar las técnicas competidoras e ir probándolas en distintos casos con ayuda de una computadora. Mientras que el enfoque *teórico (o a priori)* consiste en determinar matemáticamente la cantidad de recursos necesarios para cada uno de los algoritmos como función del tamaño de los casos considerados. Los recursos que generalmente se consideran son el tiempo de computación y el espacio de almacenamiento, siendo el primero normalmente el más importante por lo que al hablar de $eficiencia$ de un algoritmo se refiere a lo rápido que se ejecuta.

Dado un problema determinado, diferentes algoritmos podrían dedicar distintas cantidades de esfuerzo para resolverlo. El grado de esfuerzo que requiere un algoritmo para encontrar la solución de un problema es lo que se denomina complejidad computacional. El tiempo que requiere un algoritmo para resolver un problema está determinado por una función que depende del tamaño x del conjunto de datos para procesar: $f(x)$.

$f(x)$ determina el esfuerzo (en tiempo) en función del tamaño de la entrada, e interesa obtener su orden de magnitud, o sea, poder determinar con qué velocidad crece a medida que aumenta el tamaño x del conjunto de entradas. Al referirnos al orden de magnitud de un algoritmo lo hacemos con expresiones como "crece tan rápido como ...", "crece por lo menos tan rápido como ...", "crece en el orden de ...". Para representar formalmente estas sentencias, existen los símbolos O, Ω, Θ, respectivamente [35].

Cota superior asintótica O

Dada $f(x)$, que expresa la complejidad de un algoritmo en función del tamaño de las entradas, la expresión $f(x) \in \mathrm{O}(g(x))$ indica que $f(x)$ no crece más rápido que $g(x)$ (a lo sumo lo hace tan rápido como $g(x)$), como se indica en la Figura 2.3. A $g(x)$ se le denomina cota superior asintótica. $\mathrm{O}(g(x))$ es un conjunto que se puede describir de la siguiente manera:

$\mathrm{O}(g(x)) = \{f(x) : existen\ c, x_0 > 0\ tales\ que\ \forall x \geq x_0 : 0 \leq \mid f(x) \mid \leq c \mid g(x) \mid\}$

Cota inferior asintótica Ω

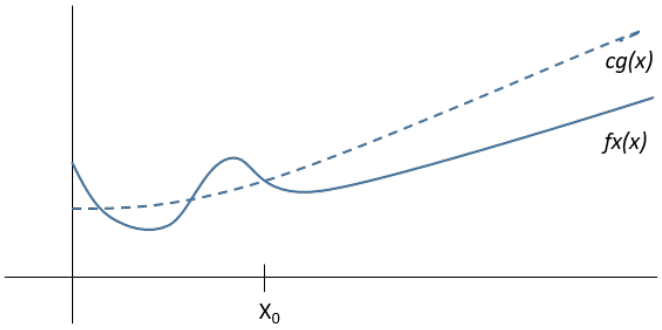


Figura 2.3: Cota superior asintótica O

De manera análoga como se muestra en la Figura 2.4, $f(x) \in \Omega(g(x))$ indica que $f(x)$ crece más rápido que $g(x)$ (al menos crece a igual velocidad que $g(x)$). En términos formales:

$\Omega(g(x)) = \{f(x) : existen\ c, x_0\ constantes\ positivas\ tales\ que\ \forall x \geq x_0 : 0 \leq | f(x) \geq c \mid g(x) |\}$

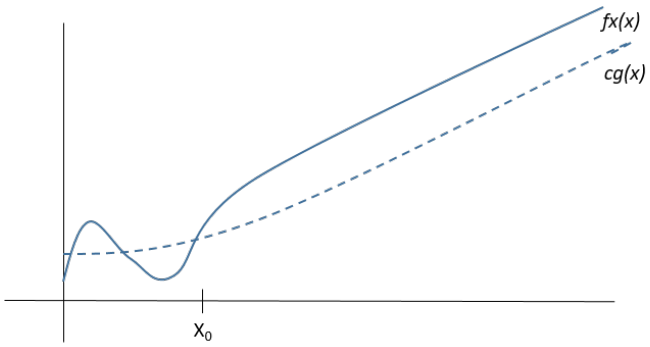


Figura 2.4: Cota inferior asintótica Ω

Cota ajustada asintótica Θ

Por último, $f(x) \in \Theta(g(x))$ indica que $f(x)$ y $g(x)$ crecen a la misma velocidad como se muestra en la Figura 2.5. Este conjunto se define como:

$\Theta(g(x)) = \{f(x) : existen\ c, d, x_0\ constantes\ positivas\ tales\ que\ \forall x : x_0 \leq x : 0 \leq c \mid g(x) | \leq | f(x) | \leq d \mid g(x) |\}$

El empleo de cotas asintóticas para comparar algoritmos tiene sentido para entradas de tamaño considerable. En ocasiones comparar algoritmos utilizando un número pequeño de datos de entrada puede conducir a conclusiones incorrectas.

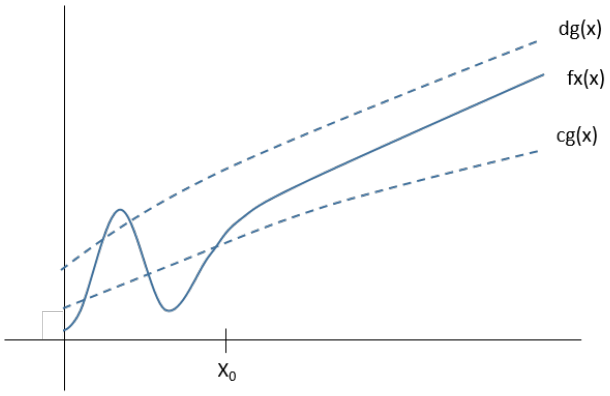


Figura 2.5: Cota ajustada asintótica Θ

2.2.2. Clases de complejidad

En la vida real existen muchos problemas prácticos para los cuales no se conoce ningún algoritmo eficiente, pero cuya dificultad intrínseca ha sido tema de muchas investigaciones. Entre estos se cuentan problemas tan conocidos como el del agente viajero, el coloreo de grafos, el problema de la mochila, los ciclos hamiltoneanos, la programación entera, la búsqueda del camino simple más largo de un grafo y la satisfactibilidad de una fórmula Booleana [90].

La importancia del orden de magnitud puede verse en la Tabla 2.1, en esta tabla observamos que el orden de complejidad crece a medida que aumenta el tamaño de la entrada n para diferentes funciones de complejidad temporal.

Tabla 2.1: Funciones de complejidad temporal

	Tamaño del problema (n)			
Función	10	10^2	10^3	10^4
$log_2 n$	3,3	6,6	10	13,3
n	10	10^2	10^3	10^4
$nlog_2 n$	0,33 x 10^2	0,7 x 10^3	10^4	1,3 x 10^5
n^2	10^2	10^4	10^6	10^8
2^n	1024	1,3 x 10^{30}	$> 10^{100}$	$> 10^{100}$
$n!$	3 x 10^6	$> 10^{100}$	$> 10^{100}$	$> 10^{100}$

Por lo general, la teoría de la complejidad está diseñada especialmente para apli-

carse a **problemas de decisión**. Un problema de decisión (PD) se plantea como una pregunta general, la cual acepta como respuesta sólo una de dos posibilidades, la respuesta "SI" o la respuesta "NO". En forma abstracta, un problema de decisión puede describirse de la siguiente forma:

$PD :< D, S >$ Dónde: D - Dominio de valores posibles y $S \subseteq D$ - son las instancias que resuelven el problema. El planteamiento del PD es:

$$\forall x \in D : PD(x) = \begin{cases} Si & \text{si x} \in \text{S} \\ No & \text{en otro caso} \end{cases}$$

Como mencionamos antes, un algoritmo es un procedimiento general que trabaja paso a paso y en un número finito de estos, resuelve un problema. Un algoritmo resuelve un problema de decisión PD si puede aplicarse a cualquier instancia l del PD y garantiza que siempre produce una solución para esa instancia. Podemos pensar en un algoritmo como un programa de computadora, o como la descripción de la función de transición de una máquina de Turing.

La ejecución de un algoritmo se expresan en términos de una sola variable: la longitud o tamaño de las instancias l del problema, que refleja la cantidad de datos de entrada y la magnitud de cada uno de estos datos.

Dada una instancia de un problema de decisión PD con m datos de entrada, $(d_1, \ldots, d_m)$, se define la longitud de la instancia como: $long(l) = \sum_{k=1}^{m} long(d_i)$, donde $long(d_i)$ es la longitud de cada uno de los datos de entrada.

La función de complejidad en tiempo de un algoritmo expresa los requerimientos de tiempo, que un determinado modelo de computación necesita al ejecutar el algoritmo para resolver cada posible instancia del problema. Con el fin de clasificar el esfuerzo computacional que se requiere al resolver los problemas de decisión, estos problemas se han clasificado en clases de complejidad. Una misma clase de complejidad contiene a todos los problemas que requieren funciones similares de tiempo para ser resueltos. Denotando a un algoritmo determinista como AD y a un algoritmo no-determinista como AnD, podemos definir las siguientes clases de complejidad:

- $DLOG = \{PD \mid \exists\ AD \textit{ que resuelve } PD \textit{ usando espacio logaritmico}\}$
- $P = \{PD \mid \exists\ AD \textit{ que resuelve } PD \textit{ en tiempo polinomial}\}$
- $NP = \{PD \mid \exists\ AnD \textit{ que resuelve } PD \textit{ en tiempo polinomial}\}$

- $EXP = \{PD \mid \exists\ AD\ que\ resuelve\ PD\ en\ tiempo\ exponencial\}$

Así el conjunto de problemas que pueden ser resueltos en tiempo polinomial por una máquina de Turing determinista se define como la clase de complejidad P, mientras que el conjunto de problemas que pueden ser resueltos en tiempo polinomial pero por una máquina de Turing no determinista es denotado como la clase de complejidad NP.

Definición 2.11 *NP es la clase de problemas de decisión que admiten un sistema de pruebas $F \subseteq X \times Q$ tal que existe un polinomio $p(n)$ y un algoritmo de tiempo polinomial A tal que:*

- *Para todo $x \in X$ tal que existe un $q \in Q$ tal que $(x, q) \in F$ y además el tamaño de q es como máximo $p(n)$, donde n es el tamaño de x.*
- *Para todos los pares (x, y) el algoritmo A puede verificar si (x, y) pertenece o no a F. En otras palabras, $F \in P$.*

La complejidad NP es la clase de problemas que tienen un algoritmo determinista de resolución que corre en tiempo exponencial, pero para los cuales también existe un algoritmo no determinista de resolución que corre en tiempo polinomial.

Las clases de complejidad son ampliamente utilizadas para clasificar una gran diversidad de problemas y a través de estas clasificaciones se han desarrollado diversas áreas de investigación. Por ejemplo, una línea crucial de investigación es encontrar problemas que inicialmente se clasifican en la clase NP, y que a través de un análisis profundo, se puede demostrar que están en la clase P.

Los problemas NP-completos son aquellos que se resuelven en tiempo polinomial mediante procesos no deterministas, mientras que un proceso determinista requiere un tiempo de orden exponencial. A los problemas NP-completos también se les conoce como problemas intratables.

Un algoritmo se define como *No Determinístico* como sigue: *un algoritmo no determinístico es un algoritmo que consta de dos fases: suponer y comprobar*. Por ejemplo, dado el problema de satisfactibilidad con una fórmula booleana particular, al principio un algoritmo no determinístico supone una asignación y luego comprueba si esa asignación satisface la fórmula o no.

El problema de implicación lógica es un problema que pertenece a la clase Co-NP, mientras que el problema de revisión de creencias es NP^{NP}, esto es, está en el segundo nivel de la jerarquía polinomial. La clase Co-NP denota el conjunto de problemas de decisión cuyo complemento está en NP. Llamamos además NP-difícil a un problema G si cualquier instancia de un problema genérico de la clase NP puede ser reducido a una instancia de G utilizando una transformación en tiempo polinomial (de igual forma aplica para la clase coNP-difícil).

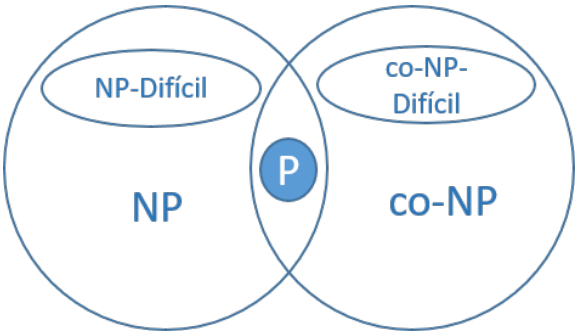


Figura 2.6: Clases de complejidad

En el diagrama de la Figura 2.6 notamos que P $\subseteq$ NP y P $\subseteq$ Co-NP. De esta forma decimos que un problema es tratable si pertenece a la clase P, y un problema es intratable si pertenece a NP o Co-NP.

De igual forma si C es una clase de complejidad de problemas de decisión, denotamos mediante $Co - C$ la clase de problemas de decisión cuyos complementos están en C. Por ejemplo, el conjunto de fórmulas Booleanas que no son satisfactibles y el conjunto de grafos que no poseen un ciclo hamiltoniano pertenecen a $Co - NP$. Está claro que P es un subconjunto tanto de NP como de $Co - NP$, y que NP y $Co - NP$ son ambos subconjuntos de $PSPACE$. $PSPACE$ es la clase de todos los problemas de decisión que se pueden resolver empleando como máximo un número polinómico de bits de almacenamiento.

Se sabe que $NP = Co - NP$ si y solo si existe un problema $NP - completo$ en $Co - NP$. No hay muchos problemas de los cuales se sepa que están en $NP \cap Co - NP$, y que sin embargo se crea que no están en P. Entre estos mencionamos el conjunto de los números primos y el problema de decisión de Turing equivalente polinómicamente a la factorización.

La importancia de los problemas NP y en partícular los $NP-Completos$ reside en que identifican un amplio tipo de problemas difíciles. Un problema difícil es aquel cuya cota inferior parece ser del orden de una función exponencial. En otras palabras, la teoría de los problemas $NP-Completos$ ha identificado un extenso tipo de problemas, que en apariencia carecen de algoritmos de tiempo polinomial que los resuelvan. Algunos problemas que se creyeron NP-difíciles terminaron siendo de complejidad polinomial, por ejemplo, el problema de la búsqueda, el problema del árbol de expansión mínimo, ambos se resuelven mediante un algoritmo polinomial, los cuales reciben el nombre de problemas de la clase P (polinomiales).

El conjunto de problemas $NP-Completos$ conocidos es muy grande y sigue aumentando. Todos estos problemas poseen una característica común: *hasta ahora, en el peor caso, ningún problema $NP-Completo$ puede resolverse con un algoritmo polinomial. En otras palabras, hasta la fecha, en el peor caso, el mejor algoritmo determinista para resolver cualquier problema $NP-Completo$ tiene complejidad exponencial.*

De acuerdo a la teoría de problemas $NP-Completos$ se cumple lo siguiente: *si un problema $NP-Completo$ puede resolverse en tiempo polinomial, entonces todos los problemas NP pueden resolverse en tiempo polinomial. O bien, si cualquier problema $NP-Completo$ puede resolverse en tiempo polinomial, entonces NP=P?.*

Así que, la teoría de los problemas $NP-Completos$ indica que todo problema $NP-Completo$ es semejante a un pilar decisivo. Si se derrumba un pilar, todo el edificio se viene abajo.

Debido a que es muy improbable que todos los problemas $NP-Completos$ puedan resolverse con algoritmos polinomiales, en consecuencia es muy improbable que cualquier problema $NP-Completo$ pueda resolverse con cualquier algoritmo polinomial.

2.2.3. Satisfactibilidad

El problema de satisfactibilidad (SAT) fue el primer problema $NP-Completo$ que se descubrió [2]. Determinar la satisfactibilidad de una fórmula proposicional es una tarea relevante con repercusiones en otros problemas estudiados en la Inteligencia Artificial (IA), tales como: en la estimación del grado de creencia, para revisar

o actualizar las creencias [26], en la explicación abductiva, en el diagnóstico lógico, entre otros procedimientos.

Definición SAT

El problema de satisfactibilidad se define como sigue: dada una fórmula Booleana, determinar si esta fórmula es satisfactible o no.

Sea F una Forma Normal Conjuntiva (FNC), esto es, una conjunción de disyunciones de literales. Se sabe que el problema SAT(F) es NP-completo a partir de tener cláusulas en F con al menos 3 literales.

Ejemplo 1. Considere el siguiente conjunto de cláusulas:

- $c_1 \rightarrow (\neg x_1 \vee \neg x_2 \vee x_3)$
- $c_2 \rightarrow (x_1 \vee x_4)$
- $c_3 \rightarrow (x_2 \vee \neg x_1)$

Es posible construir un árbol semántico como se muestra en la Figura 2.7.

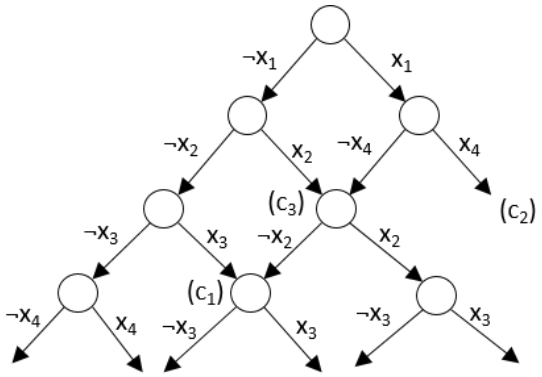


Figura 2.7: Árbol semántico

Con base en el árbol semántico de la Figura 2.7, se concluye que satisface la fórmula que contienen las tres clausulas c_1, c_2 y c_3 y por tanto es satisfactible. Todas las siguientes asignaciones satisfacen dicha fórmula:

- (x_1, x_2, x_3, x_4)
- $(x_1, x_2, x_3, \neg x_4)$

- $(\neg x_1, \neg x_2, x_3, x_4)$
- $(\neg x_1, \neg x_2, \neg x_3, x_4)$
- $(\neg x_1, x_2, x_3, x_4)$
- $(\neg x_1, x_2, \neg x_3, x_4)$

En 1960 Martin Davis y Hilary Putnam [57] desarrollaron un algoritmo para comprobar la satisfactibilidad de fórmulas de lógica proposicional en FNC. El algoritmo usa una forma de resolución en la cual las variables son elegidas iterativamente y eliminadas mediante la resolución sobre el conjunto de cláusulas de la fórmula.

En 1962 se desarrolló el algoritmo DPLL por Davis-Putnam-Logemann- Lovelandes [56], un algoritmo completo basado en el retroceso hacia atrás (back- tracking) que sirve para decidir la satisfacibilidad de las fórmulas de lógica proposicional en forma normal conjuntiva; es decir, para resolver el problema SAT, al igual que lo hacía el algoritmo anterior de Davis y Putnam.

El algoritmo de Davis y Putnam se convirtió en uno de los algoritmos clásicos para decidir la Satisfactibilidad de fórmulas proposicionales y suele ser un proceso común a aplicarse en la mayoría de los algoritmos completos, como es el caso, en los algoritmos de Satz [50], [42], [81] y [61]. Hooker [65] plantea el problema de satisfactibilidad incremental y se describe una implementación basada en el método de Davis-Putnam-Loveland para comprobar la satisfactibilidad del conjunto original de cláusulas, y considerando el caso incremental de ir adicionando nuevas cláusulas.

Con respecto a la clase de algoritmos heurísticos que son propuestos que intentan hallar soluciones de manera rápida (en tiempos polinomiales de cómputo), para el problema SAT se han aplicado búsquedas locales, tales como: GSAT y WalkSAT [18], así como algoritmos evolutivos [60] y algoritmos aleatorios [38].

Si la complejidad temporal de la etapa de comprobación de un algoritmo no determinístico es polinomial, entonces este algoritmo no determinístico se denomina algoritmo polinomial no-determinístico. Si un problema de decisión puede resolverse con un algoritmo polinomial no-determinístico, entonces este problema se denomina problema polinomial no-determinístico (NP).

La complejidad computacional asociada con la solución de problemas NP-completos, ha motivado la búsqueda de métodos alternativos que permitan la solución en tiempo polinomial de instancias especiales de problemas NP-completos. Por ejemplo, en el caso de SAT(F), si F es una 2-FNC (denotado como 2-SAT), estas instancias se resuelven mediante procesos deterministas en un tiempo polinomial. Este tipo de problemas son conocidos también como problemas tratables.

2.3. Modelos de revisión de creencias

La revisión de creencias también llamada dinámica del cambio, grado de creencias, cambio de creencias, dinámica del conocimiento, teoría de revisión de creencias, teoría del cambio de creencias o revisión del conocimiento, ha sido estudiada en base a diferentes propuestas de modelos, entre los principales se encuentran el modelo AGM y KM.

2.3.1. AGM

El modelo AGM es el modelo más utilizado para representar una teoría epistemológica de cambio de creencias, este modelo está estructurado en términos de oraciones de un lenguaje formal L de lógica proposicional. Entonces el estado epistémico de un agente (donde el "agente" refiere, indistintamente, a un ser humano, una máquina, o una institución, entre otras posibilidades) está representado por el conjunto K de oraciones de L que éste acepta. El lenguaje L posee los conectivos lógicos: negación ($\neg$), conjunción ($\wedge$), disyunción ($\vee$), implicación ($\rightarrow$), equivalencia ($\leftrightarrow$). L es cerrado respecto de estos conectivos de modo que L representa a todas las fórmulas bien formadas. En el modelo AGM el símbolo $\perp$ denota una contradicción y el símbolo $\top$ denota una tautología [26].

En el modelo AGM, las creencias de un agente están representadas como un conjunto de creencias cerrados por el operador de consecuencia lógica "Cn" que cumple con las siguientes propiedades:

- Inclusión: $X \subseteq Cn(X)$
- Monotonía: Si $X \subseteq Y$, entonces $Cn(X) \subseteq Cn(Y)$

- Iteración: $Cn(X) = Cn(Cn(X))$
- Supraclasicalidad: Si p se deriva lógicamente de X, entonces $p \in Cn(X)$
- Deducción: $q \in Cn(X \cup \{p\})$ si y solo si $(p \to q) \in Cn(X)$
- Compacidad: Si $p \in Cn(X)$, entonces $p \in Cn(X')$ para algún subconjunto finito $X' \subseteq X$

De esta manera K representa el conjunto de creencias del agente, si y solo si $K = Cn(K)$ y $K \subseteq L$.

Dentro de AGM se identifican tres criterios de racionalidad [26]:

- Primacía de la nueva información: la nueva información es siempre aceptada.
- Consistencia: el nuevo estado epistémico debe, si es posible, ser consistente.
- Economía informacional: retener cuanto sea posible de las creencias preexistentes.

El modelo AGM consiste de tres operaciones básicas como se indican a continuación:

1. Expansión: La expansión de creencias se deriva de "aprender algo". La expansión de K por ϕ se denota $K + \phi$, se cumple además los siguientes 6 postulados:

- $(E1)$ Si K es un CC (Conjunto de Creencias), $K + \phi$ es un CC
- $(E2)$ $\phi \in K + \phi$
- $(E3)$ $K \subseteq K + \phi$
- $(E4)$ Si $\phi \subset K$, entonces $K + \phi = K$
- $(E5)$ Si $K_1 \subseteq K_2$, entonces $K_1 + \phi \subseteq K_2 + \phi$
- $(E6)$ Para todo K y ϕ, $K + \phi$ es el menor CC que satisface $(E1) - (E5)$

2. Revisión: Una operación de revisión consiste según [26] en la modificación del estado epistémico. El estado epistémico es representado como un conjunto de creencias, donde una nueva creencia (la entrada epistémica) se incorpora al conjunto previo conservando consistencia lógica. $K * \phi$ denota la revisión de K por ϕ. Esta operación debe cumplir los siguientes 8 postulados:

- $(R1)$ Si K es un CC, $K * \phi$ es un CC
- $(R2)\phi \in K * \phi$
- $(R3)K * \phi \subseteq K + \phi$
- $(R4)$ Si $\neg\phi \notin K$, entonces $K * \phi = K + \phi$
- $(R5)K * \phi = K_{\perp}$ ssi $\neg\phi \in Cn(\emptyset)$
- $(R6)$ Si $\phi \leftrightarrow \psi \in Cn(\emptyset)$, entonces $K * \phi = K * \psi$
- $(R7)K * (\phi \wedge \psi) \subseteq (K * \phi) + \psi$
- $(R8)$ Si $\neg\psi \notin K * \phi$, entonces $(K * \phi) + \psi \subseteq K * (\phi \wedge \psi)$

3. Contracción: Una contracción ocurre cuando una sentencia es refutada, pero ninguna sentencia es sumada. La contracción de K por una sentencia ϕ se denota $K - \phi$, esta operación debe cumplir los siguientes 8 postulados:

- $(C1)$ Si K es un CC, $K - \phi$ es un CC
- $(C2)K - \phi \subseteq K$
- $(C3)$ Si $\phi \notin K$, entonces $K - \phi = K$
- $(C4)$ Si $\phi \notin Cn(\emptyset)$, entonces $\phi \notin K - \phi$
- $(C5)K \subseteq (K - \phi) + \phi$
- $(C6)$ Si $\phi \leftrightarrow \psi \in Cn(\emptyset)$, entonces $K - \phi = K - \psi$
- $(C7)K - \phi \cap K - \psi \subseteq K - (\phi \wedge \psi)$
- $(C8)$ Si $\phi \notin K - (\phi \wedge \psi)$, entonces $K - (\phi \wedge \psi) \subseteq K - \phi$

2.3.2. KM

Otro de los modelos base en la teoría de revisión de creencias es el modelo KM presentado en [39], los cuales unificaron los diferentes enfoques semánticos que un operador de revisión de creencias debería cumplir.

Definición 2.12 *Una función* $\circ$ *: Fórmulas* $\times$ *Fórmulas* $\rightarrow$ *Fórmulas que toma una fórmula* ψ *que representa a un conjunto de creencias, y a otra fórmula* μ *que representa la nueva información y las envía a una nueva fórmula proposicional denotada como* $\psi \circ \mu$*,* γ *representa cualquier fórmula proposicional. De esta manera* $\circ$ *es llamado operador de revisión KM.*

Un operador $\circ$ debe cumplir con los siguientes postulados:

- $(KM1)$ $\psi \circ \mu \vdash \mu$
- $(KM2)$ Si $\psi \wedge \mu$ es consistente,entonces $\psi \circ \mu \equiv \psi \wedge \mu$
- $(KM3)$ Si μ es consistente, entonces $\psi \circ \mu$ es también consistente
- $(KM4)$ Si $\psi_1 \equiv \psi_2$ y $\mu_1 \equiv \mu_2$, entonces $\psi_1 \circ \mu_1 \equiv \psi_2 \circ \mu_2$
- $(KM5)$ $(\psi \circ \mu) \wedge \gamma \vdash \psi \circ (\mu \wedge \gamma)$
- $(KM6)$ Si $(\psi \circ \mu) \wedge \gamma$ es consistente, entonces $\psi \circ (\mu \wedge \gamma) \vdash (\psi \circ \mu) \wedge \gamma$

Existen otros modelos que se han aplicado a la revisión de creencias como el modelo DP por sus autores [4] y el modelo de semi-revisión de Hanson, el cual requiere un tratamiento eficiente de inconsistencias locales, que resulta factible en conjuntos de creencias no clausurados (bases de creencias) y no en conjuntos clausurados. El método de semi-revisión de creencias es una forma de cambio de creencias que no asigna ningún valor o prioridad especial a la nueva información. De acuerdo a esto, la semi-revisión por una sentencia, es el proceso de decidir o analizar no solamente la sentencia propuesta, sino también la negación de ésta. Como resultado de la semi-revisión será que el sistema de creencias incorpore la sentencia propuesta, o incorpore la negación de la sentencia o bien que permanezca inmune al cambio propuesto.

Capítulo 3

Trabajo relacionado

En este capítulo se describen los trabajos relacionados en el campo de la teoría de revisión de creencias presentando un panorama general del problema para posteriormente mostrar el avance de dicha teoría de forma cronológica para culminar con los principales modelos y operadores que se han desarrollado hasta la fecha.

Algunos problemas computacionalmente difíciles, por ejemplo, la deducción en bases de conocimiento lógico, son tales que parte de una instancia se conoce mucho antes que el resto, y permanece igual para varias instancias posteriores del problema. En estos casos, es útil preprocesar fuera de línea esta parte conocida para simplificar el problema

El problema de revisión de creencias por su naturaleza como un problema difícil ha sido atacado desde diferentes perspectivas y diferentes teorías. Por ejemplo, la deducción en bases de conocimiento lógico, son tales que parte de una instancia se conoce mucho antes que el resto, y permanece igual para varias instancias posteriores del problema. En estos casos, es útil preprocesar fuera de línea esta parte conocida para simplificar el problema [54]. Algunas instancias de los problemas NP-difícil al realizar preprocesamiento producen problemas simplificados en tiempo polinomial, mientras que para otros implica algunas consecuencias que se consideran poco probables de poder reducir la complejidad del problema. En [45] se explora una clase de operadores de actualización de creencias en la cual la definición del operador es compuesto con respecto a la nueva información que se agregará. El objetivo es proporcionar un operador de actualización que sea intuitivo, en el sentido de que su

definición se basa en una descomposición recursiva de la estructura de la información de actualización, y que pueda ser razonablemente implementado.

3.1. Revisión de creencias con lógica proposicional

Una forma de abordar el problema de revisión de creencias es modelarlo utilizando lógica proposicional. En [31] se describe la relación directa del problema de implicación proposicional entre formas normales que puede reducirse al problema clásico Co-NP-completo de revisar la tautologicidad de una fórmula disyuntiva. Aunque la inferencia proposicional sea un problema Co-NPcompleto en su versión general, también existen diferentes casos que pueden resolverse de manera eficiente. Cuando se dispone de nueva información, los conjuntos respectivos se modifican de forma diferente para preservar partes del conocimiento durante el proceso de revisión. Este enfoque permite tratar escenarios difíciles y complejos [59], pero si la nueva información es fiable nos puede ahorrar tiempo de procesamiento, en [84] se propone que cuando la nueva información proviene de otro agente, primero debemos determinar si ese agente debe ser confiable, se define la confianza como un paso previo al procesamiento de la revisión, se hace hincapié en que la confianza en un agente a menudo se restringe a un dominio particular. Liberatore en [78] y [82] propone realizar la revisión de creencias considerando la plausibilidad de la información, dada una secuencia de revisiones, junto con sus resultados, para derivar un posible orden inicial que los ha generado.

En [83] se especifica que la nueva información debe ser confiable antes de revisar y actualizar la base de conocimiento. Esto es aplicable si se trata de una base de creencias o múltiples creencias. En creencias múltiples la fusión de creencias tiene como objetivo combinar piezas de información de diferentes (y posiblemente conflictivas) fuentes a fin de producir una única base de creencias. La base de creencias resultante debe ser consistente con la mayor cantidad de información posible.

En nuestro trabajo tomamos como base la inferencia proposicional para modelar el proceso de revisión de creencias, la lógica ha sido motivo de estudio en el tema de revisión de creencias por varios autores. En [53] se presenta un modelo teórico abstracto caracterizando los postulados AGM y KM para diferentes lógicas entre las

que se encuentran: Lógica Proposicional ($PropositionalLogic(PL)$), Lógica de Horn ($HornLogic(HCL)$), Lógica de Primer Orden ($FirstOrderLogic(FOL)$) y Lógica Descriptiva ($Descriptionlogic(DL)$) en términos de cambio mínimo entre interpretaciones. En [64] se presenta una investigación del comportamiento de los operadores de actualización y revisión considerando los postulados KM sobre lógica proposicional en fragmentos de Horn, Krom y afines, se destaca las diferencias entre revisión y actualización en este contexto.

En [1] se hace hincapié en que la representación del conocimiento y el razonamiento utilizando la lógica proposicional es un componente importante de los sistemas de inteligencia artificial. Una fórmula proposicional en forma normal conjuntiva (CNF) puede contener cláusulas redundantes, cláusulas cuya eliminación de la fórmula no afecta al conjunto de sus modelos. La identificación de cláusulas redundantes es importante porque la redundancia a menudo conduce a cálculos innecesarios, almacenamiento desperdiciado y puede oscurecer la estructura del problema. Los resultados experimentales revelan que muchas instancias de CNF obtenidas de las aplicaciones prácticas de SAT exhiben un alto grado de redundancia. De igual forma en [24] se propone un algoritmo basado en la descomposición para problemas de revisión en la lógica proposicional clásica. Un conjunto de reglas de descomposición se presentan para analizar la satisfactibilidad de las fórmulas considerando un conjunto de literales. Se construye una función de descomposición para calcular todos los conjuntos de literales satisfactorios de una fórmula dada ya que representan todos los modelos que satisfacen a la fórmula.

Otro tipo de lógica utilizada en el estudio de operadores de revisión y contracción de AGM es utilizando las teorías lógicas de entrada / salida (*input/output logical theories*) como lo indica [30]. En dicha propuesta reemplazan las fórmulas proposicionales en el marco AGM de cambio de teoría por pares de fórmulas proposicionales, que representan el carácter basado en reglas de las teorías, y reemplazan además el operador de consecuencia clásico Cn por una lógica de entrada / salida. De igual forma en [12] se estudia una familia general de operadores de actualización de creencias en un entorno proposicional. Sus operadores se basan en la dependencia fórmula/literal, que es más refinada que la noción de dependencia de fórmula/variable. La dependencia de fórmula/variable es un caso particular de dependencia de fórmu-

la/literal. Los autores argumentan que esto permite manejar el problema del marco y la ramificación de una manera más apropiada.

Existen también propuestas de algoritmos de aproximación como en [92] donde se plantea una forma de resolver el problema de la complejidad computacional de la revisión de la base de conocimiento proposicional introduciendo algoritmos de aproximación. Las soluciones propuestas satisfacen los postulados racionales de AGM, mientras que las complejidades de tiempo de los algoritmos aproximados están en un nivel más bajo que el tiempo complejidades de los enfoques existentes en las literaturas, aunque pueden no generar la solución óptima.

3.2. Desarrollo de la teoría de revisión de creencias

La teoría de revisión de creencias consiste básicamente en investigar el cambio en las bases de conocimiento, actualmente existe gran variedad de propuestas, usualmente cada método, modelo u operador describe un conjunto de axiomas o postulados a cumplir.

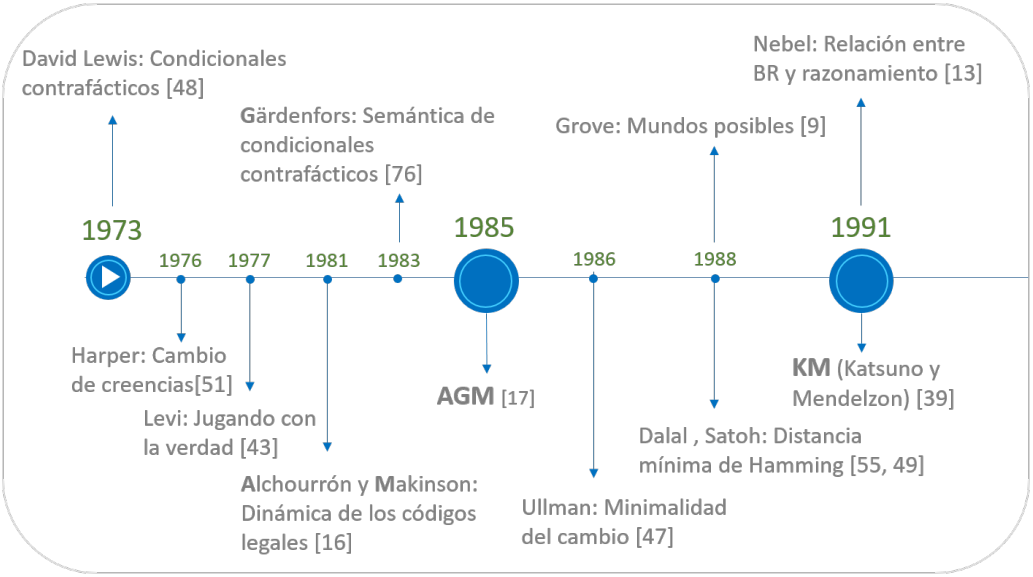


Figura 3.1: Desarrollo de la teoría de revisión de creencias 1973 - 1991

En la Figura 3.1 se describe el desarrollo de la teoría de revisión de creencias de 1973 a 1991, la mayoría de los autores hacen referencia a los trabajos de [17] con el modelo AGM, existen trabajos anteriores como son [48] y [51] sobre condicionales contrafácticos (un condicional contrafáctico es aquel $A \rightarrow B$ cuyo antecedente A es falso).

En [43] se presenta un amplio debate sobre el problema de la creencia racional desarrollada sobre el modelo de teoría de la decisión bayesiana. Posteriormente [16] tenían como objetivo formalizar la dinámica de los códigos legales. Mientras que [76] estaba interesado en establecer una semántica para los condicionales contrafácticos.

En [26] describe que Gärdenfors siendo editor de la revista Theoria recibe el artículo de Alchourrón y Makinson y descubre que estaban trabajando en los mismos problemas formales, aunque desde diferentes ópticas, por lo que deciden unir esfuerzos y así surge en 1985 la propuesta del modelo que lleva sus iniciales, el modelo AGM, el cual ha sido el origen de esta teoría de revisión de creencias.

En [9] se presenta un modelo alternativo para las funciones de cambio basado en un sistema de esferas para los "mundos posibles", este modelo provee una semántica para el modelo AGM que permite capturar cierta noción de correctitud y completitud. En [47] se propone el principio de minimalidad del cambio, este principio establece que la base de conocimiento debería cambiar lo mínimo posible cuando se incorpore nueva información.

Diversos autores como Dalal, Satoh, Wislett, Borguida y Forbus propusieron métodos de revisión de creencias aplicando la distancia mínima de Hamming como se muestra en [55], [49] y [80]. En 1991 aparece el modelo KM por sus autores [39] que proponen la unificación de los diferentes enfoques semánticos que un operador de revisión de creencias debería cumplir a través de 6 postulados. Por su parte [13] mostró una estrecha correspondencia de revisión de creencias y razonamiento.

En la Figura 3.2 mostramos el principal desarrollo de la teoría de revisión de creencias entre los años 1992-2010. En [96] se presenta un primer análisis de la complejidad del problema de revisión de creencias. En [26] se indica que posterior a la propuesta AGM, Alchourrón y Makinson desarrollaron un modelo constructivo para funciones de cambio llamado "contracción segura" que después fue generalizada por [67] donde le llamó contracción del núcleo y estaba basada en una selección

entre las sentencias de un conjunto de creencias K que contribuyen efectivamente a implicar p; y usar esta selección para contraer K por p. En [25] se propone que cada observación es una sentencia general que se asume es consistente.

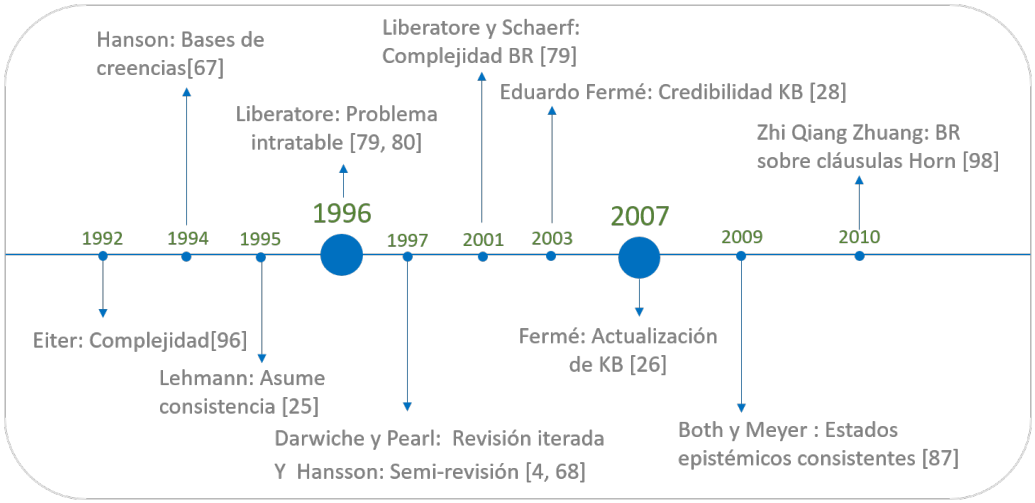


Figura 3.2: Desarrollo de la teoría de revisión de creencias 1992 - 2010

Como se ha visto en [96] un problema importante es explorar la complejidad computacional de los métodos de revisión de creencias de bases de conocimiento. Los métodos que se conocen hasta ahora son intratables en el caso general por lo que es relevante encontrar bajo que restricciones algunos métodos pudieran ser tratables [79] y [80].

En 1997 aparecieron dos modelos más, el primero denominado DP por sus autores [4], los cuales propusieron postulados para una revisión de forma iterada, donde caracterizan la revisión de creencias como un proceso que puede depender de elementos de un estado que no necesariamente son capturados por un conjunto de creencias. El segundo modelo es una aportación de [68], el cual plantea un modelo alternativo a AGM, llamado semi-revisión. Este difiere respecto del modelo estándar en que la sentencia que provoca una revisión no siempre es aceptada,

En [28] proponen representar el conocimiento a través de bases de creencias en lugar de teorías lógicas y aquellos en los que el objeto del cambio epistemológico no tiene prioridad sobre la información existente como es el caso en el modelo AGM. En [87] consideraron operadores de revisión con estados epistémicos conservando consistencia sobre lógica de Horn. En [26] se mostró la relación entre el modelo

AGM y la lógica condicional para construir funciones de cambio en la actualización de la base de conocimiento, mientras que en [98] se propone la actualización de la base de conocimiento en lógica de Horn.

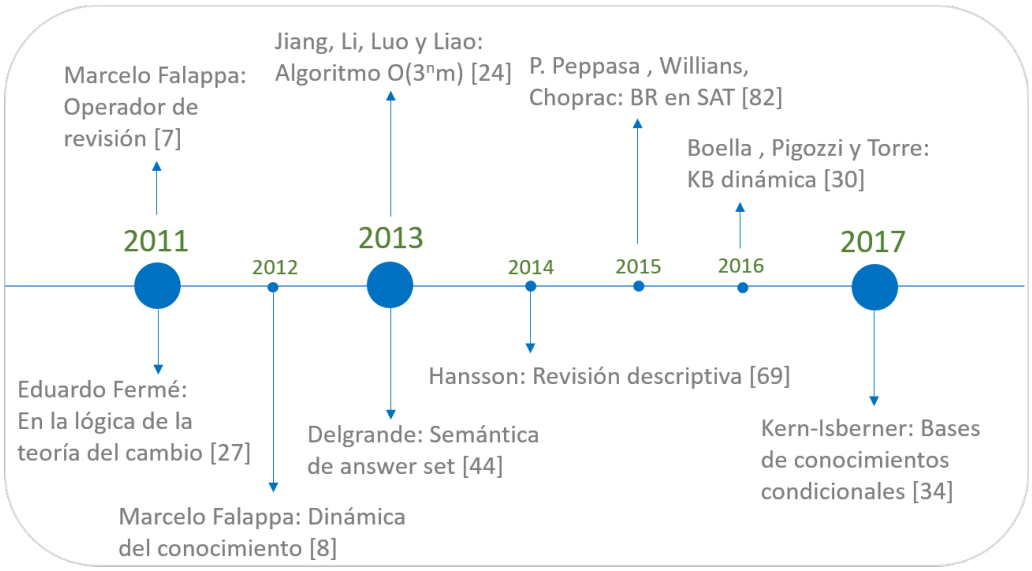


Figura 3.3: Desarrollo de la teoría de revisión de creencias 2011 - 2017

En la Figura 3.3 se describe los principales autores en el desarrollo de la teoría de revisión de creencias. En el trabajo de tesis doctoral de [27] se plantea una extensión del modelo AGM en la representación del problema de revisión de creencias utilizando otros modelos como: probabilísticos, clasificación del agente de modelos que representan el grado de creencia, el lenguaje de la lógica modal y la lógica condicional. En [7, 8] se hace una propuesta de un operador de revisión de creencias y una descripción del desarrollo de la teoría de la dinámica del conocimiento.

También han habido intentos de trabajar con otro tipo de lógica como la semántica en programación lógica con el enfoque de answer-set (expresar un problema con un conjunto de reglas lógicas) por ejemplo en [44].

En [24] se presenta el análisis de la complejidad del proceso de revisión de creencias con un orden de $O(3^n m)$, donde n es el número de fórmulas y m es el total de literales.

En [69] se propone un teorema de credibilidad del enfoque AGM más general al que se llamó revisión descriptiva como un caso especial de la revisión de creencias.

Mientras que en [82] reducen el problema de una operación de cambio de creencias para el problema de satisfactibilidad.

En [30] utilizan entradas y salidas de teorías lógicas y revisión de creencias dinámicas con el operador de contracción de Levi.

En el trabajo [19] se estudia la transformación de los sistemas de bases de conocimientos condicionales, que permiten identificar y eliminar los condicionales innecesarios de la base de conocimientos y en [34] se proponen algunos nuevos postulados para la revisión iterada múltiple que van en concordancia a los postulados de AGM y al modelo [4].

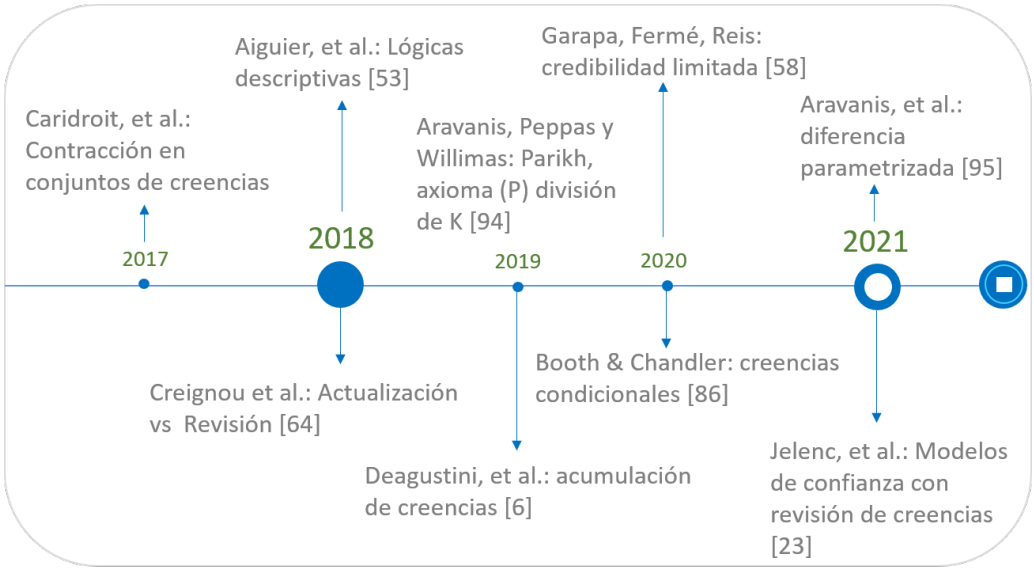


Figura 3.4: Trabajos actuales en la teoría de revisíon de creencias

En la Figura 3.4 de la revisión de los trabajos relacionados con el presente trabajo de tesis, se indica que las axiomatizaciones clásicas de la revisión de creencias incluyen postulados [11]. Los cuales establecen que si la nueva información es consistente con las creencias iniciales, entonces la revisión es simplemente equivalente a agregar la nueva información a la base de conocimientos original.

Recientemente en [53] se muestra la caracterización de varios tipos de lógicas en el estudio de la revisión de creencias, entre las cuales se tienen: lógica proposicional, lógica de Horn, lógica de primer orden y lógica de descripción, todas estas en términos de cambio mínimo entre interpretaciones. Mientras que [64] se presenta

una investigación que detalla el comportamiento de operadores de actualización y de revisión, donde consideran cláusulas de Horn y Krom.

Un nuevo operador denominado operador de revisión de diferencia parametrizada (DP) [95] que caracteriza la resistencia al cambio epistémico, este tipo de operadores DP es una generalización del operador de revisión de Dalal [97], con un rango de aplicabilidad mucho mayor. En [86] se presenta un fortalecimiento más modesto de la propuesta de Darwiche y Pearl. Mientras que los postulados de DP restringen la relación entre un conjunto de creencias condicionales anterior y posterior, los nuevos principios gobiernan la relación entre dos conjuntos de creencias condicionales posteriores obtenidos de un previo común mediante diferentes revisiones. Otro tipo de operador se presenta en [58], conocido como operador de revisión de base de credibilidad limitada, que aborda dos de los principales problemas que se han señalado al modelo AGM de cambio de creencias. Por un lado, estos operadores se definen en base a creencias (en lugar de conjuntos de creencias) y, por otro lado, se construyen con la idea subyacente de que no se acepta toda la información nueva.

En el desarrollo de la revisión de creencias, una propuesta diferente se realiza en [94], donde se proporcionan las caracterizaciones del atrincheramiento epistémico y del encuentro parcial del axioma sensible a la relevancia de Parikh para la revisión de creencias, conocido como axioma (P). El axioma (P) establece que, si un conjunto de creencias K se puede dividir en dos compartimentos separados y la nueva información se relaciona solo con el primer compartimento, entonces la revisión de K no debería afectar al segundo compartimento. En [6] se presenta un enfoque novedoso para la contracción de bases de conocimiento donde las fórmulas tienen valores adjuntos que miden alguna calidad vinculada a esas fórmulas, explotándola para definir su deseabilidad y utiliza dicha deseabilidad para definir qué fórmulas deben eliminarse para resolver conflictos sin examinar toda la base de conocimiento. El enfoque se basa en el uso de la acumulación de creencias donde varias fórmulas utilizan de manera colaborativa sus respectivos valores para prevalecer en la resolución de conflictos. Esta misma teoría de contracción de bases de conocimiento se aplican a la teoría de conjuntos de creencias [88].

La aplicación de la teoría de creencias se ha extendido en los modelos de confianza que se han vuelto invaluables en escenarios dinámicos, como las aplicaciones

de Internet, ya que brindan medios para estimar la confiabilidad de las contrapartes de interacción potenciales. En [23] se propone un modelo de confianza donde se utilizan comparaciones por pares como calificaciones y donde la confianza se expresa como un estricto orden parcial inducido sobre los agentes. Para mantener un ordenamiento sólido, el modelo utiliza una técnica de revisión de creencias que evita las contradicciones que pueden surgir al agregar nueva información. La técnica utiliza mecanismos que razonan cuantitativamente sobre la confiabilidad de la información, lo que permite al modelo descontar calificaciones en el tiempo y resistir el engaño.

3.3. Modelos y operadores de revisión de creencias

Del estado del arte obtenemos como resumen (ver Tabla 3.1) los principales modelos de revisión de creencias. En el trabajo de tesis utilizamos el modelo KM debido a que, para KM un estado epistémico es una fórmula proposicional de un lenguaje proposicional finito que codifica un conjunto de creencias.

Tabla 3.1: Principales modelos de revisión de creencias

Modelos	**Operaciones**	**Postulados**	**Características**
AGM [17]	Expansión Revisión Contracción	6 8 8	Modelo base
Mundos posibles [9]	Revisión	-	Probabilístico
Levi [43] y Harper	Expansión Revisión Contracción	2	Refomularon AGM
Dalal [55]	Revisión	6	Distancia Hamming entre cláusulas
KM [39]	Revisión	6	Modelo semántico
DP [4]	Revisión	6	Revisión iterada
Harper [51]	Contracción	2	Contracciíon iterada
Semi Revisión [68]	Revisión	6	p y $\neg p$

La combinación de operadores representa una herramienta importante para extraer una vista coherente e informativa de un conjunto de agentes. La consideración

de escenarios prácticos a incrementar sustancialmente el interés en desarrollar sistemas que trabajen con modelos de confianza. En este contexto, [5] propone una aproximación al problema de la fusión de conocimientos en un escenario multiagente donde cada agente asigna a otros agentes un valor que refleja su percepción sobre la credibilidad de cada agente, de esta forma se introduce un operador para fusionar ontologías de Datalog considerando la credibilidad de los agentes.

El estudio del cambio de creencias iterado se ha centrado principalmente en la revisión, y el otro operador principal de la teoría del cambio de creencias de AGM, esto es, la contracción, la cual ha recibido relativamente poca atención. En [85] los principios de la revisión iterada se pueden trasladar a la contracción iterada generalizando un principio conocido como la "Identidad de Harper". La identidad de Harper proporciona una receta para definir el conjunto de creencias resultante de la contracción de una oración A en términos de (i) el conjunto de creencias inicial y (ii) el conjunto de creencias resultante de la revisión por $\neg A$. Aquí se buscan formas de definir de manera similar el conjunto de creencias condicionales que resulta de la contracción por A.

En [10] y [63] se presenta un operador de fusión de creencias como una operación central dentro del campo del cambio de creencias y aborda el problema de combinar múltiples bases de conocimiento, posiblemente mutuamente inconsistentes, en una única y consistente base. Una tendencia de investigación actual en el cambio de creencias se refiere a los teoremas de representación adaptados a fragmentos de lógica, en particular la lógica de Horn.

Tabla 3.2: Principales operadores de revisión de creencias

Autor	**Tipo**	**Características**
Ullman(1986) [47]	Actualización	Cambio mínimo
Ferme(2007) [26]	Revisión y Actualización	Teoría del cambio
Falapa(2011) [8]	Revisión	Actualización de la KB en Agentes
Falapa(2011) [27]	Revisión	Revisíon en dinámica del conocimiento
Haret (2017) [10]	Fusión	Cambio de creencias
Creignpu(2018) [64]	Actualización	Cambio de creencias
Booth(2019) [85]	Contraccíon	Contracción iterada
Garapa(2020) [58]	Credibilidad Limitada	Revisión generalizada
Aravanis(2021) [95]	Diferencia Parametrizada	Revisión

En la tabla 3.2 se resumen los principales operadores desarrollados para la revisión de creencias clasificados por año y por autor, en particular [24] es uno de los que reporta la complejidad computacional del operador desarrollado en los artículos revisados a la fecha.

Capítulo 4

Metodología del proyecto de tesis

En este capítulo se describe la metodología del proyecto de tesis, el diseño del algoritmo de revisión de creencias y los procedimientos utilizados para tal proposito.

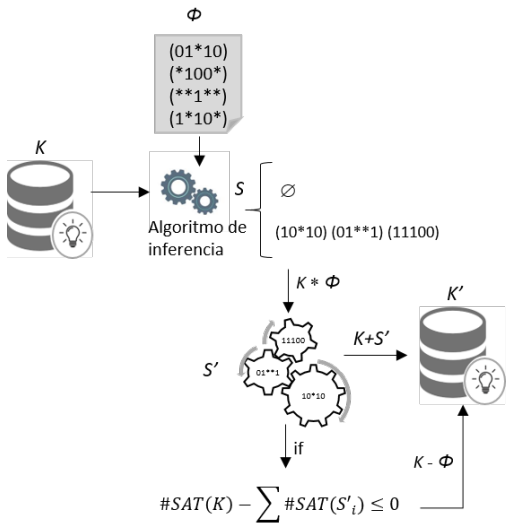


Figura 4.1: modelo base de inferencia proposicional para actualizar la base de conocimiento K.

La Figura 4.1 representa el modelo general utilizado en el diseño de los algoritmos propuestos. El diagrama se describe paso a paso:

- K representa la base de conocimiento.
- ϕ representa la nueva información que se revisa para ser añadida a K.
- $K * \phi$ representa el proceso de revisión de ϕ vs K (propuesta del algoritmo de inferencia), este proceso permite determinar si la nueva información se agrega o no a K, es decir, verificar si $K \models \phi$.
- Como resultado del proceso de revisión obtenemos un conjunto S. Si $S = \emptyset$ entonces $K \models \phi$.
- En caso contrario, es decir si $S \neq \emptyset$ entonces verificamos el número de modelos de $K' = K + S$ (propuesta algorítmica de conteo de modelos).
- Si $\#Sat(K') > 0$, entonces actualizamos $K = K'$.
- En caso contrario, Si $\#Sat(K') = 0$ significa que la base de conocimiento no es consistente, por lo que es necesario realizar un proceso de contracción. El proceso de contracción se representa como $K - C_j$, con C_j en K y consisten en eliminar la creencia más antigua de la base de conocimiento K (propuesta estrategía de contracción, obteniendo S').

Para la generación de la propuesta de un algoritmo de revisión de creencias usando inferencia proposicional se consideraron las siguientes etapas:

4.1. Etapas del diseño de los algoritmos

- Etapa 1. Revisar el estado del arte del campo de revisión de creencias, en particular el modelo AGM y KM. Así como el estado del arte de la inferencia proposicional aplicada a la revisión de creencias.
- Etapa 2. Diseñar una propuesta algorítmica de un operador de revisión de creencias.
- Etapa 3. Generar un conjunto de instancias de prueba con lógica proposicional que sirvan como datos de entrada a la propuesta de solución del algoritmo propuesto de revisión de creencias.

- Etapa 4. Realizar el análisis de la complejidad en tiempo del algoritmo propuesto.

Etapa 1. Revisar la teoría de revisión de creencias

En la etapa 1 que consiste en la revisión del estado del arte, se realizó una revisión desde los inicios de la teoría de revisión de creencias hasta el año actual, en el capítulo 3 se describe dicha investigación. Cabe resaltar que a pesar que existen diversos modelos de revisión de creencias se sigue considerando el modelo AGM y KM como referente en el diseño de nuevas propuestas de operadores para la revisión de creencias. En esta primera etapa también se realizó una revisión de propuestas usando inferencia proposicional, donde se ha aplicado operadores de revisión de creencias sobre cláusulas de Horn y Krom [64]. Como resultado de esta etapa inicial de la metodología tenemos las siguientes consideraciones:

- El proceso de inferencia proposicional puede ser modelado utilizando lógica proposicional.

- El modelo base para formalizar un operador de revisión de creencias puede ser el modelo KM ya que representa la unificación de los principales postulados del modelo AGM.

- A pesar que el problema general de la inferencia proposicional es Co-NP Completo ([40]), existen también algunos casos que se pueden resolver de manera eficiente ([89]) si consideramos una KB $K = \bigwedge_{(j=1)}^{m} C_j$ y $\phi = \bigwedge_{(i=1)}^{k} \phi_i$ donde cada $C_j \in K$ y cada $\phi_i \in \phi$ son cláusulas expresadas bajo un mismo conjunto de n variables booleanas.

Etapa 2. Diseño del algoritmo

En la etapa 2 se realizó una propuesta de un algoritmo de revisión de creencias que se describe en la sección 4.2. Para dicha propuesta algorítmica de inferencia proposicional se tienen las siguientes consideraciones:

- Se utiliza la forma normal conjuntiva para representar tanto la KB K como la nueva información ϕ a través de un conjunto de cláusulas falsificantes.

- Dadas dos FNC's F_1 y F_2 decir si $F_1 \models F_2$ es lógicamente equivalente a probar que $\neg F_2 \models \neg F_1$ que resulta de revisar la inferencia entre formas disyuntivas, ya que si F es una FNC entonces $\neg(F)$ es una FND. Como K y ϕ están en FNC, las cadenas falsificantes de sus cláusulas $Fals(k)$ y $Fals(\phi)$ se pueden calcular eficientemente [31].

- Para el diseño del algoritmo propuesto se propone el método de recorrido a lo profundo de forma recursiva lo que nos permite crear un árbol de deducción al comparar cada cláusula del conjunto de la nueva información ϕ con la base de conocimiento K.

- Se propone una estrategía para verificar la consistencia de la base de datos y realizar un proceso de contracción de K.

- Se desarrolló una propuesta para el conteo de modelos de la base de conocimiento K con el fin de determinar si es consistente(satisfactible) o no.

Etapa 3. Casos de prueba

En la etapa 3 se ha trabajado en un conjunto de casos de prueba. Las fórmulas se crean en FNC con máximo 10 variables con el fin de ejemplificar la funcionalidad del algoritmo propuesto. Se utilizan archivos de texto plano como formato para los archivos de entrada de la base de conocimiento y de la nueva información. Se generaron dos aplicaciones web para realizar pruebas con más variables.

Etapa 4. Cálculo de la complejidad del algoritmo En la etapa 4 sobre el cálculo de la complejidad del algoritmo propuesto se utilizará la notación O (O-grande) como medida para representar el orden de crecimiento.

4.2. Diseño del algoritmo de revisión de creencias

Sea K una FNC, es decir, $K = \bigwedge_{i=1}^{m} C_i$, donde cada $C_i, i = 1, \cdots, m$ es una disyunción de literales. El conjunto de asignaciones que forman $Fals(C_i)$ se puede representar mediante una cadena A_i del conjunto $\{0, 1, *\}$. Si $C_i = \{x_{i1} \vee \cdots \vee x_{ik}\} \in$

K, entonces el valor para cada posición de $i_1 - th$ a $i_k - th$ de la cadena A_i, tiene que falsificar las literales de C_i. Si $x_{ij} \in C_i$, entonces el $i_j - th$ ésimo elemento de A_i se establece en 0. Si $\neg xi_j \in C_i$, entonces el $i_j - th$ ésimo elemento de A_i se establece en 1. Las variables en $v(K)$ que no aparecen en C_i se representan por el símbolo *, lo que significa que podrían tomar cualquier valor lógico $\{0, 1\}$. De esta forma, la cadena A_i de longitud $n = |v(K)|$ representa el conjunto de asignaciones que falsifican la cláusula C_i [72].

Denotaremos la cadena que representa $Fals(C_i)$ como $A_i = cadena(Fals(C_i))$. Es fácil construir $Fals(K)$ ya que cada cláusula C_i determina un subconjunto de asignaciones de K. De esta forma, expresaremos $Fals(C)$ como la cadena falsificadora o como el conjunto de literales (cláusulas) de manera indistinta, ya que ambas expresiones denotan el conjunto $Fals(C)$. El siguiente lema expresa cómo formar el conjunto de asignaciones falsificadoras de un FNC.

Lema 1. Dado una FNC $K = \bigwedge_{i=1}^{m} C_i$, se cumple que, $Fals(K) = \bigcup_{i=1}^{m} \{\sigma \in S(K) | Fals(C_i) \subseteq \sigma\}$.

Si dos cláusulas C_i y C_j que tienen un par de literales complementarias, entonces, sus asignaciones falsificantes tambien tienen literales complementarias, esto es, $Fals(C_i) \cap Fals(C_j) = \emptyset$.

En el diseño de las propuestas algorítmicas se utilizó un conjunto de rutinas para comparar y generar cláusulas, a continuación se describen estas rutinas en pseudocódigo y se dan ejemplos de su funcionamiento.

La revisión de creencias vista como inferencia proposicional tiene los siguientes hechos: Sea K una base de conocimiento y sea ϕ la nueva información:

- $K \models \phi$ Si SAT(K) $\subseteq$ SAT(ϕ)
- $K \models \phi$ Si Fals(ϕ) $\subseteq$ Fals (K)

El proceso de revisión de creencias utilizando inferencia proposicional se basa en representar la base de conocimiento mediante una fórmula compuesta de un conjunto de cláusulas en forma normal conjuntiva. Dicha fórmula se transforma (como se indica en la Figura 4.2) a un conjunto de cadenas falsificantes que representarán a Fals(K), utilizando un patrón de ceros y unos y para el caso de que una variable no aparezca, entonces, se asigna un asterisco "*". El Ejemplo 4.1 muestra dicha trasformación.

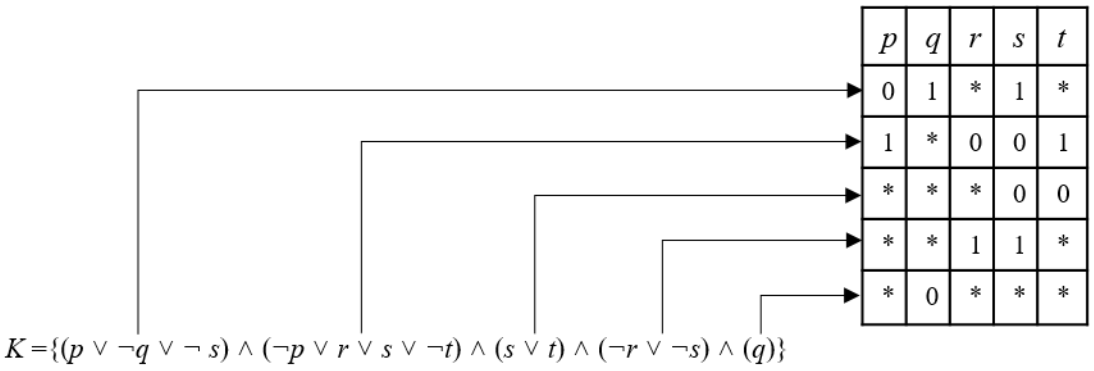


Figura 4.2: Patrón de transformación fasificante de una fórmula en FNC a 0,1 y *

Ejemplo 4.1 *Representando una fórmula en FNC usando patrones falsificantes. Sea K y ϕ las siguientes FNC's:*
$K = \{(\neg p \vee r) \wedge (q \vee r \vee \neg s) \wedge (p) \wedge (\neg q \vee \neg s)\}$ *y* $\phi = \{(\neg p \vee s) \wedge (q \vee r \vee \neg s \vee t) \wedge (\neg p \vee q \vee \neg s) \wedge (\neg r)\}$.

Reescribimos ambas fórmulas considerando el espacio de variables: p, q, r, s, t, como patrones de ceros, unos y asteriscos para representar el espacio Fals(K) y Fals(ϕ).
$K = \{(1 * 0 * *), (*001*), (0 * * * *), (*1 * 1*)\}$
$\phi = \{(1 * *0*), (*0010), (10 * 1*), (* * 1 * *)\}$

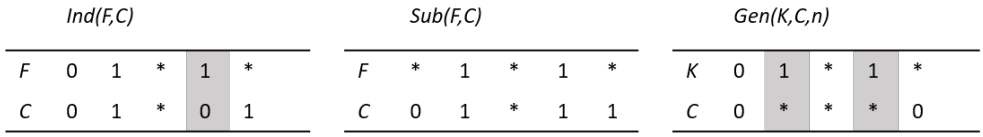


Figura 4.3: Operaciones básicas en el proceso de revisión de creencias

Además de transformar la base de conocimiento y la nueva información, en el proceso general de la propuesta algorítmica para la revisión de creencias utilizamos las operaciones básicas (ver Figura 4.3). Donde F y C representan cláusulas en el mismo conjunto de n variables trasformadas usando el patron de ceros, unos y asterisco.

- $Ind(F, C)$**:** dos cláusulas (F, C) son independientes si tienen variables contrarias para una misma posición en dichas cláusulas.

- $Sub(F, C)$**:** una cláusula C es subsumida por una cláusula F si los modelos de C están contenidos en los modelos de F, es decir, al comparar las cláusulas C y F, cada literal en $C_i = F_i$ o bien $C_i = 0, 1$ y $F_i = *$ con i=1 hasta n, donde $n = |C|$. Si una cláusula es subsumida significa que dicha cláusula se puede inferir lógicamente de la base de conocimiento, por lo que el número de modelos no cambia.

- $Dif(F, C)$**:** obtiene la diferencia $\mid Lit(F_i) - Lit(C_i) \mid$ entre las literales de una cláusula F y una cláusula C.

- $Gen(F, C, n)$**:** dadas F y C dos cláusulas, se generan n cláusulas si en la posición $C_i = *$ y F_i es una literal con valor de {0,1}. Donde $n = \mid Dif(F, C) \mid$ que determina el número de nuevas cláusulas a generar.

A continuación se decriben cada uno de los subprocesos involucrados en el proceso general del diseño de los algoritmos propuestos.

El Algoritmo 1 recibe como entrada una cláusula C_i de la base de conocimiento K y una cláusula F_j de ϕ con el fin de verificar si alguna literal de F_i esta negada en C_i devolviendo "true" si las cláusulas son independientes o "'false" en caso contrario. En el Ejemplo 4.2 se muestra la aplicación de la función $Ind(F, C)$ que permite verificar la propiedad de independencia entre dos cláusulas de forma algorítmica.

Algoritmo 1 Función $Ind(F, C)$

```
para i ← 1 hasta sizeof(C) hacer
  si (F_i = 0 & C_i = 1) | (F_i = 1 & C_i = 0) entonces
    retornar true
  fin si
fin para
retornar false
```

Ejemplo 4.2 *Sea K y ϕ el siguiente conjunto de cláusulas: $K = \{C_1, C_2, C_3, C_4\}$ = {(10*0*), (*110*), (*101*),(110*0) } y $\phi = \{F_1\} = \{(11 * *1)\}$. Aplicando el Algoritmo 1 sobre cada cláusulas de K obtenemos los resultados:*

$Ind(F_1, C_1) = true$, dado que en la posición dos de F_1 contienen un 1 y en la posición dos de C_1 hay un 0.
$Ind(F_1, C_2) = false$, debido a que no hay variables con valores opuestos (dependientes).
$Ind(F_1, C_3) = false$, debido a que son cláusulas dependientes.
$Ind(F_1, C_4) = true$, son independientes dado que en la última posición de F_1 hay un 1 mientras que en la última posición de C_4 hay un 0.

Algoritmo 2 Función $Dif(F, C)$

```
K_dif ← 0
para i ← 1 hasta sizeof(C) hacer
  si (F_i = * & (C_i = 1 | C_i = 0)) entonces
    K_dif ← K_dif + 1
  fin si
fin para
retornar K_dif
```

El Algoritmo 2 $Dif(F, C)$ recibe como entrada una cláusula F del conjunto de la nueva información ϕ y una cláusula C de la base de conocimiento K. Y la salida es la diferencia (valor numérico) de ceros y unos entre estas dos cláusulas. El valor resultante determina el número de nuevas cláusulas que pueden ser generadas. Si la salida de la función es cero (0) significa que la cláusula F es subsumida por C o que ambas cláusulas son independientes. El ejemplo 4.3 muestra la aplicación de la función $Dif(F, C)$.

Ejemplo 4.3 *Sea $K = \{C_1, C_2, C_3, C_4, C_5\} = \{$(10*0*), (*110*), (*101*), (110**), (10**0)$\}$ y $\phi = \{(11 * *1)\}$. Los resultados de la aplicación de la función $Dif(F, C)$ son:*
$Dif(F_1, C_1) = 1$.
$Dif(F_1, C_2) = 2$.
$Dif(F_1, C_3) = 2$.
$Dif(F_1, C_4) = 1$.
$Dif(F_1, C_5) = 0$.

Algoritmo 3 Function $Gen(F, C, n)$

```
para j ← 1 hasta n hacer
  para i ← 1 hasta sizeof(C) hacer
    si (F_i = * & C_i = 1) entonces
      F_i ← 0
    sino
      si (F_i =* & C_i = 0) entonces
        F_i ← 1
      fin si
    fin si
  fin para
  Push(φ,F)
fin para
```

El Algoritmo 3 $Gen(F, C, n)$ genera al menos n cláusulas de acuerdo a la diferencia obtenida por el algoritmo 2. Cada nueva cláusula representa el espacio de posibles soluciones de acuerdo al número de variables. El Ejemplo 4.4 muestra la generación de nuevas cláusulas.

Ejemplo 4.4 *Generando nuevas cláusulas.*
*Sea F = (1***1) una cláusula de ϕ, y sea C = (*110*) una cláusula de K. Sea $n =\mid Dif(F, C) \mid$. Como n = 3, $Gen(F, C, 3)$ generan 3 nuevas cláusulas:* $\{(10 * *1), (110 * 1), (11111)\}$.

Los procedimientos antes descritos serán utilizandos en las diferentes propuestas algorítmicas que se describen en la siguiente sección.

4.3. Algoritmo de revisión de creencias

En general los operadores de revisión de creencias, como Dalal y Satoh, requieren calcular el conjunto de modelos; $Sat(K)$ y $Sat(\phi)$, que equivale a realizarse en un procedimiento exponencial. Dado que determinar $Sat(K)$ es NP-completo. En este caso, en lugar de calcular $Sat(K)$ y $Sat(\phi)$, nuestra propuesta del operador de revisión trabaja sobre los conjuntos $Fals(K)$ y $Fals(\phi)$. Nos basamos en el uso de patrones falsificantes que permiten calcular los conjuntos $Fals(K)$ y $Fals(\phi)$ en

tiempo lineal sobre el tamaño de K y ϕ. Esos conjuntos se utilizan para determinar si ϕ se infiere de K. Una de las principales tareas a realizar en el proceso de revisión de creencias consiste en comprobar si $K \models \phi$. Si conocemos $Fals(K)$ y $Fals(\phi)$, entonces $(Fals(\phi) \subseteq Fals(K))$ si y solo si $K \models \phi$.

Definición 4.1 *Sea K una base de conocimiento (KB) y sea ϕ la nueva información, ambas fórmulas proposicionales expresadas en forma normal conjuntiva (FNC). $K \circ \phi$ denota la propuesta algorítmica para realizar la revisión de creencias de ϕ en K.*

La propuesta para el algoritmo de revisión de creencias está respaldada por un proceso de construcción de $(Fals(\phi) - Fals(K))$. Si éste último conjunto está vacío, entonces $K \models \phi$, y en este caso $K \circ \phi = K$. De lo contrario, sea $S = (Fals(\phi) - Fals(K))$ el conjunto de asignaciones que falsifican a ϕ, los cuales no falsifican a K. Cuando construimos los conjuntos en FNC tales que $Fals(F_s) = S$, entonces hemos encontrado la FNC F_s, tales que $(K = K \wedge F_s)$ determinan que $K \models \phi$.

En este algoritmo de revisión de creencias destacamos los siguientes elementos:

- Proponemos un método que funciona en el conjunto de falsificar asignaciones de las fórmulas involucradas para revisar $K \models \phi$.
- Introducimos un operador lógico entre dos cláusulas, denotadas $Ind(\phi_i, C_j)$, y eso construye una nueva FNC, tal que $Fals(F_s) = Fals(\phi_i) - Fals(C_j)$.
- Esta propuesta de revisión de creencias de ϕ en K, denotado $(K \circ \phi)$, mantiene los postulados de Katsuno y Mendelzon (KM).
- Se implementó el operador $Ind(\phi_i, C_j)$ para trabajar en tiempo lineal en el número de variables involucradas, y es la base de nuestro procedimiento para la revisión de creencias.
- Aunque el operador $Ind(\phi_i, C_j)$ se ejecuta eficientemente, se trata de una aplicación iterativa sobre cada $C_j \in K$ para generar nuevas F_s tales que $(K \wedge F_s) \models \phi$, por lo que la aplicación iterada de $Ind(\phi_i, C_j)$ puede generar un crecimiento exponencial del número de cláusulas en F_s.

4.3.1. Construcción del conjunto independiente de cláusulas

Como K y ϕ son FNC, las asignaciones falsificantes $Fals(K)$ y $Fals(\phi)$ se pueden calcular de forma eficiente. Revisar $K \models \phi$ es equivalente a comprobar si $Sat(K) \subseteq Sat(\phi)$, es decir, $Fals(\phi) \subseteq Fals(K)$. El resultado de aplicar el operador de revisión de creencias entre un KB K y la nueva evidencia ϕ se denota por $K' = K \circ \phi$.

Cuando $K \models \phi$, entonces definimos $K \circ \phi = K$, porque la nueva evidencia se infiere de K, y entonces K no necesita cambiar. Si $K \not\models \phi$ entonces $Fals(\phi) \not\subset Fals(K)$, lo que implica que hay un conjunto de asignaciones S tal que $S \subseteq Fals(\phi)$ y $S \not\subseteq Fals(K)$ [32].

Cuando $K \not\models \phi$, nuestro método de revisión de creencias funciona construyendo $S = (Fals(\phi) - Fals(K)) \neq \emptyset$ lo que permite construir un nuevo FNC F_s, tal que $S = Fals(F_s)$ y $K = (K \wedge F_s)$, y se mantiene $K \models \phi$. $K' = (K \wedge \phi)$ tiene menos información que K (porque K' tiene más cláusulas que K). De hecho, si $S \neq \emptyset$ entonces $Fals(K) \subset Fals(K')$, y por lo tanto, $Sat(K') \subset Sat(K)$.

Observe que la falsificación de cadenas para cláusulas independientes tiene valores complementarios (0 y 1) en al menos uno de sus valores fijos. Sea $C_i \in K$ y $x \in v(K) \setminus v(C_i)$ cualquier variable, tenemos que:

$$C_i \equiv (C_i \vee \neg x) \wedge (C_i \vee x) \tag{4.1}$$

Además, esta reducción conserva el número de asignaciones falsificantes de C_i. Dado que $\#Fals(C_i) = 2^{n-|C_i|} = 2^{n-(|C_i|+1)} + 2^{n-(|C_i|+1)} = \#Fals((C_i \vee \neg x) \wedge (C_i \vee x))$, debido a que $(C_i \vee \neg x)$ y $(C_i \vee x)$ son dos cláusulas independientes.

Definición 4.2 *Dado un par de cláusulas dependientes C_1 y C_2, Si $Lit(C_1) \subseteq Lit(C_2)$ decimos que C_2 es subsumida por C_1.*

Si C_1 subsume a C_2, entonces $Fals(C_2) \subseteq Fals(C_1)$. Por otro lado, si C_2 no está subsumido por C_1 y son dependientes, existe un conjunto de índices $I = \{1, \ldots, p\} \subseteq \{1, \ldots, n\}$, tal que para cada $i \in I$, $x_i \in C_1$ pero $x_i \notin C_2$. Existe una reducción para transformar C_2 para que sea independiente con C_1, llamamos a esta trasformación reducción de independencia entre dos cláusulas. La reducción de independencia funciona de la siguiente manera: sean C_1 y C_2 dos cláusulas dependientes y sea $\{x_1, x_2, \ldots, x_p\} = Lit(C_1) \setminus Lit(C_2)$. Por 4.1 podemos escribir:

$C_1 \wedge C_2 = C_1 \wedge (C_2 \vee \neg x_1) \wedge (C_2 \vee x_1)$. Ahora C_1 y $(C_2 \vee \neg x_1)$ por 4.1 son independientes. Aplicando 4.1 a $(C_2 \vee x_1)$:

$C_1 \wedge C_2 \equiv C_1 \wedge (C_2 \vee \neg x_1) \wedge (C_2 \vee x_1 \vee \neg x_2) \wedge (C_2 \vee x_1 \vee x_2)$

Las tres primeras cláusulas son independientes con C_1. Si repetimos el proceso de independizar la última cláusula de las anteriores hasta que se considere x_p, tenemos que $C_1 \wedge C_2$ se puede escribir como: $C_1 \wedge (C_2 \vee \neg x_1) \wedge (C_2 \vee x_1 \vee \neg x_2) \wedge \ldots \wedge (C_2 \vee x_1 \vee x_2 \vee \ldots \vee \neg x_p) \wedge (C_2 \vee x_1 \vee x_2 \vee \ldots \vee x_p)$.

La última cláusula contiene todos los literales de C_1, por lo que está subsumida por C_1, y entonces:

$$C_1 \wedge C_2 \equiv C_1 \wedge (C_2 \vee \neg x_1) \wedge (C_2 \vee x_1 \vee \neg x_2) \wedge \ldots \wedge (C_2 \vee x_1 \vee x_2 \vee \ldots \vee \neg x_p). \quad (4.2)$$

Obtenemos del lado derecho de un conjunto independiente de cláusulas $p + 1$, que denotamos como $ind\_red(C_1, C_2)$. Usaremos la reducción independiente entre dos cláusulas C_1 y φ (o entre sus respectivas cadenas falsificantes) para definir:

$$Ind(\varphi, C) = \begin{cases} \varphi & Si\ \varphi\ y\ C\ son\ independientes \\ \emptyset & Si\ Lit(C) \setminus Lit(\varphi) = \emptyset \\ ind\_red(C, \varphi) - C & En\ otro\ caso \end{cases} \quad (4.3)$$

Es fácil redefinir el operador $Ind(\varphi, C)$ en términos de las cadenas falsificantes que representan a $Fals(C)$ y $Fals(\varphi)$. La operación $Ind(\varphi, C)$ forma una conjunción de cláusulas cuyas asignaciones falsificantes son exactamente $Fals(\varphi) - Fals(C)$.

4.3.2. Propuesta algorítmica

En este caso, en lugar de calcular el conjunto de modelos de las fórmulas, nuestro operador de revisión trabaja sobre los conjuntos de asignaciones falsas de las fórmulas involucradas. El método de revisión de creencias se basa en las siguientes propiedades:

1. Si $\forall s \in Fals(\phi)$, se cumple que, $s \in Fals(K)$, entonces, $K \models \phi$.

2. Si $\exists s \in Fals(\phi)$, y $s \notin Fals(K)$, entonces $K \not\models \phi$.

El primer caso considera que $Fals(\phi) \subseteq Fals(K)$, lo que demuestra que $K \models \phi$. Para este caso $K \circ \phi = K$, ya que KB K no necesita cambiar. En el segundo caso, $S = Ind(\phi, K)$ contiene el conjunto de asignaciones falsificantes de ϕ y $S \nsubseteq Fals(K)$.

La operación $Ind(\varphi_i, C_j)$ forma un conjunto de cadenas que representa el conjunto de asignaciones falsificantes: $Fals(\varphi_i) - Fals(C_j)$. Es decir, $Ind(\varphi_i, C_j)$ determina las asignaciones que están en $Fals(\varphi_i)$, pero que no están contenidas en $Fals(C_j)$. Cuando se aplica el operador $Ind(\varphi_i, C_j)$ sobre todo $C_j \in K$, se obtiene un conjunto S de cadenas. Además, S cumple que $S \subseteq Fals(\varphi_i)$ y $S \nsubseteq Fals(K)$.

El conjunto S permite construir una FNC Fs_i, tal que $Fs_i = (D_1 \wedge D_2 \wedge \cdots D_t)$, y $S = Fals(Fs_i)$. Cuando se agregan las nuevas cláusulas Fs_i a K, se obtiene un nuevo KB K'_i es tal que $K'_i \models \varphi_i$, además de que K'_i se mantiene como una FNC. El pseudocódigo para el procedimiento $Ind(\varphi_i, K)$ se presenta en el algoritmo 4.

Algoritmo 4 *Revision*(φ_i, K)

Entrada: K una KB y φ_i cláusulas con la nueva información
Salida: F_s un conjunto de cláusulas en FNC
Push(φ_i, V); $Fs = \{\varphi\}$;
para todo $C_j \in K$ **hacer**
  mientras $(V \neq \emptyset)$ **hacer**
    φ = Pop(V); {Obtiene la siguiente cláusula}
    $Fs = Fs - \{\varphi\}$; {Remueve la cláusula de salida}
    $Nc = Ind(\varphi, C_j)$; {Forma: $Nc \wedge C_j \models \varphi$ }
    si $(Nc \neq \emptyset)$ **entonces**
      $Fs = Fs \cup Nc$; {Sólo si hay cláusulas que añadir}
    fin si
  fin mientras
  para todo $\varphi \in Fs$ **hacer**
    Push(φ, V); {Siguiente iteración considerando nuevas cláusulas}
  fin para
fin para
retornar (Fs)

En el algoritmo *Revision*() se implementa el procedimiento $Ind(\varphi_i, K)$ que consta de dos ciclos. El ciclo más externo se ejecuta en $C_j \in K$, y es de orden $O(|K|)$. Este ciclo externo (el *for* en el algoritmo 4) itera en las columnas de una tabla, donde se almacenan los resultados de cada $Ind(\varphi_i, C_j), C_j \in K$. El cuerpo del ciclo interno consiste esencialmente en realizar el operador $Ind(\phi_i, C_j)$, que es de orden $O(n)$.

El ciclo interno genera las filas de la tabla. Una fila inicial puede expandirse para contener varias cadenas, cuando se usa una nueva cláusula C_j en $Revision()$. El número de cadenas formadas por $Ind(\varphi_i, K)$ determina el número total de cadenas asignadas a esa fila en la tabla. En el peor de los casos, $Ind(\varphi_i, K)$ puede generar un número exponencial de cadenas en n, como se muestra en la sección de análisis de la complejidad.

Cuando el proceso $Ind(\varphi_i, K)$ itera sobre todo $\varphi_i \in \phi$, se forman nuevas cláusulas Fs_i, tales que $Fals(\cup(Fs_i)) = Fals(\phi) - Fals(K)$. Cuando el conjunto de cláusulas $(\cup Fs_i)$ se agrega a K, se forma una nueva KB K'. Además, $K' \models \phi$, debido a que $Fals(\phi) \subseteq Fals(K)$. Por lo tanto, $Sat(K) \subseteq Sat(\phi)$.

Dadas dos cláusulas C_i, C_j que difieren en el signo de solo una variable, la reducción por literal complementario genera una cláusula única de $C_i \wedge C_j$. De hecho, la reducción se basará en la aplicación de la Ecuación 4.1. Por ejemplo, sea $C_i = (x \vee q)$, y $C_j = (\neg x \vee q)$, entonces $C_i \wedge C_j = (q)$. En términos de las cláusulas de falsificación de cadenas, denotamos dicha reducción como $Varcom(A_i, A_j)$.

Una estrategia relevante mientras se realiza $Revision(\varphi_i, K)$, es ordenar las cláusulas $C_i \in K$ según el tamaño $|Lit(C_j) - Lit(\varphi_i)|$. Debido a que el número de literales C_j diferente con φ_i determina el número de cláusulas independientes a generar. Además, después de aplicar $Ind(\varphi_i, K)$, es relevante aplicar la reducción por literales complementarios en las cadenas en S, así como eliminar cualquier cláusula subsumida en F_s, con el fin de minimizar el número total de cláusulas que forman a F_s.

La propuesta algorítmica aquí presentada trabaja sobre el conjunto de cláusulas construidas por $Ind(\varphi_i, K)$, que se agregarán al KB K original con el propósito de que cada $\varphi_i \in \phi$ se infiera de $K \wedge Ind(\varphi_i, K)$. Esto se muestra en el siguiente teorema.

Teorema 4.1 *Dadas dos cláusulas φ_i y C_j, se cumple que $(C_j \wedge Ind(\varphi_i, C_j)) \models \varphi_i$.*

Prueba 4.1 *Si φ_i y C_j son cláusulas independientes, entonces $\varphi_i = Ind(\varphi_i, C_j)$ y además $(C_j \wedge Ind(\varphi_i, C_j)) = (C_j \wedge \varphi_i)$. Entonces, $(C_j \wedge \varphi_i) \models \varphi_i$ debido a que $(p \wedge q) \supset q$, y por reflexividad: $\varphi_i \models \varphi_i$.*

Si φ_i y C_j no son independientes, pero $Ind(\varphi_i, C_j) = \emptyset$, entonces esto implica que, $Fals(\varphi_i) \subseteq Fals(C_j)$ y que $C_j \models \varphi_i$. Como $C_j = (C_j \wedge Ind(\varphi_i, C_j))$, entonces $(C_j \wedge Ind(\varphi_i, C_j)) \models \varphi_i$.

Cuando φ_i y C_j no son independientes, y también $Ind(\varphi_i, C_j) \neq \emptyset$, se cumple que $(C_j \wedge Ind(\varphi_i, C_j)) \equiv (C_j \wedge \varphi_i)$ por (4.2), cumpliendo que $(C_j \wedge \varphi_i) \models \varphi_i$, porque $(p \wedge q) \supset q$, y por reflexividad: $\varphi_i \models \varphi_i$. Por lo tanto, para cualquiera de los tres resultados posibles de $Ind(\varphi_i, C_j)$ obtenemos que $(C_j \wedge Ind(\varphi_i, C_j)) \models \varphi_i$ se cumple.

El conjunto de cláusulas construído usando $Ind(\varphi_i, C_j)$ contienen exactamente las cláusulas necesarias que permiten inferir cada $\varphi_i \in \phi$ a partir de $(C_j \wedge Ind(\varphi_i, C_j))$.

Ahora, podemos mostrar que el conjunto F_s de cláusulas formadas por $Ind(\varphi_i, C_j)$ representa el conjunto mínimo de cláusulas que permiten cubrir el espacio: $Fals(\varphi_i) - Fals(C_j)$, dado que, $Fals(Fs) = Fals(\varphi_i) - Fals(C_j)$. De esta forma, $(C_j \wedge Ind(\varphi_i, C_j)) \models \varphi_i$.

Teorema 4.2
$Fals(Ind(\varphi_i, C_j)) = Fals(\varphi_i) - Fals(C_j)$.

Prueba 4.2 *Si $Ind(\varphi_i, C_j) = \emptyset$, se cumple que $Ind(\varphi_i, C_j)$ es el número mínimo de cláusulas que nos permiten inferir $(C_j \wedge Ind(\varphi_i, C_j)) \models \varphi_i$, ya que de hecho, $C_j \models \varphi_i$. Supongamos ahora que $Ind(\varphi_i, C_j) \neq \emptyset$.*

Demostramos que $\forall s \in Fals(Ind(\varphi_i, C_j))$, $s \in Fals(\varphi_i)$ y $s \notin Fals(C_j)$. Sea $s \in Fals(Ind(\varphi_i, C_j))$, entonces s debe falsificar φ_i. Cada cláusula en $Ind(\varphi_i, C_j)$ tiene la forma $(\varphi_i \vee R)$, con R una disyunción de literales. Si s falsifica $(\varphi_i \vee R)$, entonces s falsifica tanto a (φ_i) como a (R), y por tanto $s \in Fals(\varphi_i)$.

Además, $s \notin Fals(C_j)$ porque C_j es independiente con cada una de las cláusulas de $Ind(\varphi_i, C_j)$ (por el operador de independencia Ind_red), y por tanto, $s \notin Fals(C_j)$.

El teorema anterior muestra que $Ind(\varphi_i, C_j)$ construye un conjunto de cláusulas que cubren exactamente las asignaciones de espacio necesarias para cumplir con $Fals(\varphi_i) \subseteq Fals(C_j) \cup Fals(Ind(\varphi_i, C_j))$.

Aún más, el conjunto $Fals(Ind(\varphi_i, C_j))$ es el conjunto mínimo de asignaciones para cubrir el espacio $Fals(\varphi_i) - Fals(C_j)$, ya que $Fals(C_j)$ y $Fals(Ind(\varphi_i, C_j))$ son ajenos (por construcción del operador de independencia), y por tanto $Fals(C_j) \cap Fals(Ind(\varphi_i, C_j)) = \emptyset$.

Corolario 4.1 $Fals(Ind(\varphi_i, K)) \subseteq Fals(\varphi_i)$.

Prueba 4.3 *$Fals(Ind(\varphi_i, C_j)) = Fals(\varphi_i) - Fals(C_j)$ (por el Teorema 4.2). Iterando sobre cada $C_j \in K$, obtenemos que $Fals(Ind(\varphi_i, K)) = Fals(\varphi_i) - Fals(K)$. Y por las propiedades entre conjuntos, se cumple que $Fals(Ind(\varphi_i, K)) \subseteq Fals(\varphi_i)$.*

Al iterar $Ind(\varphi_i, C_j)$ sobre todo $C_j \in K$, se obtiene un conjunto mínimo de cláusulas: Fs_i que asegura que: $(K \wedge Fs_i) \models \varphi_i$.

Al extender K con las cláusulas obtenidas en $Ind(\varphi_i, K)$ se va formando K'. Así K' extiende al conjunto de cláusulas de K, y por tanto, extiende también el espacio inicial de falsificaciones de K, agregando las asignaciones que falsifican a $Ind(\varphi_i, K)$. De hecho, estos dos conjuntos de falsificaciones son excluyentes por construcción de $Ind(\varphi_i, K)$, y por tanto, $Fals(K) \cap Fals(Ind(\varphi_i, K)) = \emptyset$. En otras palabras, el conjunto de modelos de K' es ahora un subconjunto de los modelos de K, $Sat(K') \subseteq Sat(K)$.

Sin embargo, al iterar el operador $Ind(\varphi_i, K)$, sobre cada $\varphi_i \in \phi, i = 1, \ldots, m$, se tiene que los K conjuntos de cláusulas Fs_i formados por $Ind(\phi, K)$ podrían no tener un número mínimo de cláusulas. La reducción $Varcom$ permite reducir el número de cláusulas entre los diferentes conjuntos $Ind(\varphi_i, K)$.

Así, después de obtener el conjunto de cláusulas $Ind(\phi, K)$, se reduce su cardinalidad, eliminando cláusulas subsumidas y aplicando la reducción $Varcom$ entre cláusulas de dos diferentes conjuntos $Ind(\varphi_{i_1}, K)$ e $Ind(\varphi_{i_2}, K)$, y así garantizar obtener en $Ind(\phi, K)$ un conjunto mínimo de cláusulas.

Este último proceso de reducción de cláusulas a través de literales complementarias y de eliminación de cláusulas subsumidas, se ejecuta en tiempo polinomial (de hecho en tiempo cuadrático) sobre la longitud inicial de $|Ind(\phi, K)|$, ya que consistiría en ir tomando una cláusula $C \in Ind(\phi, K)$, y revisar si es subcláusula (como subconjunto de literales) o si hay una literal complementaria con alguna otra cláusula en $Ind(\phi, K) - C$.

4.3.3. Aplicación del algoritmo

Mostraremos algunos ejemplos de aplicación del algoritmo de revisión de creencias.

Ejemplo 4.5 *Sea una KB $K = (\neg q \vee \neg s) \wedge (\neg p \vee \neg r \vee s) \wedge (q \vee r \vee \neg s) \wedge (p) \wedge (\neg p \vee$*

$q \vee \neg r \vee \neg s)$ *y sea* $\phi = (\neg p \vee q \vee \neg s) \wedge (\neg q \vee r \vee \neg s) \wedge (\neg p \vee s) \wedge (\neg p)$. *Tanto* K *como* ϕ *se transforman usando patrones falsificantes de 0, 1 y *. Entonces aplicando el algoritmo de revisión (Algoritmo 4), tenemos que verificar que* $K \models \phi$ *es equivalente a verificar* $Fals(\phi) = \{$*10*1, *101, 1**0, 1****$\} \subseteq Fals(K) = \{$**1*1, 1*10, *001, 0***, 1011*$\}$. *En cada celda a partir de la columna 2 de la Tabla 4.1, se muestra el resultado de* $Ind(\varphi_i, C_j)$.

Tabla 4.1: Aplicación del operador de independencia $Ind(\phi, K)$

	KB K					
ϕ	*1*1	1*10	*001	0***	1011	S
10*1	10*1	10*1	1011	1011	$\emptyset$	$\emptyset$
*101	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1**0	1**0	1*00	1*00	1*00	1*00	1*00
1***	10**	100*	1000	1000	1000	1000
		1011	1011	1011	$\emptyset$	$\emptyset$
	11*0	1100	1100	1100	1100	1100

En la Tabla 4.1 cuando revisamos φ_1=10*1 respecto a K en la penúltima columna obtenemos $\emptyset$ debido a que es una cláusula subsumida. Para φ_4=1*** se generan nuevas cláusulas con el objetivo de lograr la independencia entre cláusulas.

Las cadenas resultantes del Ejemplo 4.5 son representadas en el conjunto S={1*00, 1000, 1100}. Transformando S a sus respectivas cláusulas: $F_S = \{(\neg p \vee r \vee s) \wedge (\neg p \vee q \vee r \vee s) \wedge (\neg p \vee \neg q \vee r \vee s)\}$. De esta forma, la nueva KB $K' = K \wedge F_s$ cumple que $K' \models \phi$. Note que la cláusula 1*00 subsume a las otras dos, así que por el último proceso de simplificación S ={1*00}.

Ejemplo 4.6 *En este ejemplo se utiliza el Códice Romanoff. Se trata de una colección de notas inéditas de Leonardo Davinci, referentes a su curiosidad y su creación en torno a la cocina, la mesa del comedor y las costumbres de su época. Este es una especia de chiste perfectamente elaborado, y se ha convertido en uno de los juegos más simpáticos de la literatura gastronómica. "Estos son (algunos de) los hábitos indecorosos que los invitados a la mesa de mi señor que no deben cultivar (y esto*

basado en mi observación de aquellos que frecuentaron la mesa de mi señor durante el último año)". Aquí se muestran algunas reglas derivadas de las oraciones:

1. *Ningún invitado tiene que sentarse en la mesa o sobre el invitado*
2. *Si no levanta los pies entonces puede estar en la mesa*
3. *No debe limpiar su armadura sobre la mesa*
4. *No prender fuego a la mesa*
5. *Tampoco tiene que prenderle fuego a los invitados mientras permanecen en la mesa*

Las restricciones del Ejemplo 4.6 se modelan en lógica proposicional como: $p =$ en la mesa, $q =$ sentarse, $r =$ levantar los pies, $s =$ prender fuego, $t =$ sobre el invitado, $u =$ limpiar armadura. Las reglas que se obtienen de las frases son las siguientes:

1. $q \supset (\neg p \wedge \neg t)$
2. $\neg r \supset p$
3. $u \supset \neg p$
4. $p \supset \neg s$
5. $(p \wedge t) \supset \neg s$

Estas reglas forman la KB K. Convirtiendo K en FNC, $K = (\neg q \vee \neg p) \wedge (\neg q \vee \neg t) \wedge (r \vee p) \wedge (\neg u \vee \neg p) \wedge (\neg p \vee \neg s) \wedge (\neg p \vee \neg t \vee \neg s)$. Sea ϕ la nueva creencia $\phi = (s \wedge \neg p) \vee (s \wedge t) \equiv (s) \wedge (\neg p \vee t)$, que podría interpretarse como: prender fuego y no sobre la mesa o prender fuego y sobre el invitado. Entonces, verificar $K \models \phi$ es equivalente a verificar si $Fals(\phi) =$ {***0**, 1***0*,} $\subseteq Fals(K) =$ {11****, *1**1*, 0*0***,1****1,1**1**,1**11*}. En la Tabla 4.2 se muestra el resultado de aplicar $Ind(\varphi_i, C_j)$.

El conjunto S de cadenas resultantes de $Ind(\phi, K)$ es $S = \{0010**, 01100*, 10*0*0, 10*000\}$. y su FNC asociada es $F_s = (p \vee q \vee \neg r \vee s) \wedge (p \vee \neg q \vee \neg r \vee s \vee t) \wedge (\neg p \vee q \vee s \vee u) \wedge (\neg p \vee q \vee s \vee t \vee u)$. Mediante la aplicación de la reducción de Varcom y

también la reducción mediante cláusulas subsumidas, tenemos que $(\neg p \vee q \vee s \vee t \vee u)$ está subsumida por $(\neg p \vee q \vee s \vee u)$. Así, F_s se reduce a $(p \vee q \vee \neg r \vee s) \wedge (p \vee \neg q \vee \neg r \vee s \vee t) \wedge (\neg p \vee q \vee s \vee u)$, y $K' = K \wedge F_s = (\neg q \vee \neg p) \wedge (\neg q \vee \neg t) \wedge (r \vee p) \wedge (\neg u \vee \neg p) \wedge (\neg p \vee \neg s) \wedge (\neg p \vee \neg t \vee \neg s) \wedge (p \vee q \vee \neg r \vee s) \wedge (p \vee \neg q \vee \neg r \vee s \vee t) \wedge (\neg p \vee q \vee s \vee u)$.

Podemos asignar la siguiente interpretación a las nuevas cláusulas F_s. $(p \vee q \vee \neg r \vee s)$ es $(r \supset \neg(\neg p \wedge \neg q \wedge \neg s))$, y esto significa: Si alguien está levantando los pies, entonces no es posible que no esté sentado, no está prendiendo fuego, y no está limpiando armaduras. Si bien $(\neg p \vee q \vee s \vee u)$ es $(p \supset \neg(\neg q \wedge \neg s \wedge \neg u))$ eso significa no sentarse, no prender fuego, no limpiar la armadura sobre la mesa. $(p \vee \neg q \vee \neg r \vee s \vee t)$ es $(q \supset \neg r) \supset \neg(\neg p \wedge \neg s \wedge \neg t)$ significa: si se sienta entonces no tiene que levantar los pies, no sobre la mesa y no prender fuego y no sobre el invitado.

Tabla 4.2: Operador de independencia $Ind(\phi, k)$ para algunas reglas del Códice Romanoff

	KB k						
ϕ	11****	*1**1*	0*0***	1****1	1**1**	1**11*	S
***0**	0**0**	00*0**	0010**	0010**	0010**	0010**	0010**
		01*00*	01100*	01100*	01100*	01100*	01100*
	10*0**	10*0**	10*0**	10*0*0	10*0*0	10*0*0	10*0*0
1***0*	10**0*	10**0*	10**0*	10**00	10*000	10*000	10*000

4.3.4. Análisis de la complejidad del algoritmo

La función de tiempo para nuestro operador de revisión de creencias $(K \circ \phi)$ se denotará por $T_0(|\phi|, |K|)$. Esta función depende principalmente del tiempo de ejecución del operador de independencia: $Ind(\phi, K)$. Observe que $Ind(\phi, K)$ se obtiene del cálculo iterativo de $Ind(\varphi_i, K)$. La complejidad de tiempo para el proceso $Ind(\phi_i, K)$ es de orden $O(|K| \cdot n \cdot f(|\varphi_i|, |K|))$, donde $f(|\varphi_i|, |K|)$ es una función entera, que dada una cláusula φ_i y una FNC K, cuenta el número de cláusulas que formará $Ind(\varphi_i, K)$.

Analicemos el número máximo posible de cláusulas que se pueden generar mediante $Ind(\varphi_i, K)$. Para algunos casos, $Ind(\varphi_i, K)$ genera el conjunto vacío (cuando

$\exists\ C_j \in K$, tal que $C_j \models \varphi_i$). Sin embargo, en el peor de los casos, la complejidad temporal del cálculo de $Ind(\varphi_i, K)$ depende de la longitud de los conjuntos: $S_{ij} = \{x_1, x_2, \ldots, x_P\} = Lit(C_j)$ - $Lit(\varphi_i)$, $j = 1, \ldots, m$.

Como se indicó anteriormente, dado $\varphi_i \in \phi$, es relevante ordenar las cláusulas $Cj \in K$ de acuerdo con la cardinalidad de los conjuntos S_{ij}, $j = 1, \ldots, m$ de menor a mayor, y eliminando las cláusulas que son independientes con φ_i. Cuando no hay cláusulas independientes con φ_i, ni $S_{ij} = \emptyset$ para $j = 1, \ldots, m$, la complejidad de tiempo para calcular $Ind(\varphi_i, K))$ está limitada por el número de cláusulas resultantes. En otra palabras, $|Ind(\varphi_i, K)| \leq |S_{i1}| * |S_{i2}| * \ldots * |S_{is}| * Poly(n, m), s \leq m$. Donde $Poly(n, m)$ resume un tiempo polinomial debido al proceso de coincidencia de cadenas y la clasificación de las cláusulas en K.

Entonces, inferimos un límite superior para la función $f(|\varphi_i|, |K|)$, dado por:

$$Max\{\mid S_{i1} \mid * \mid S_{i2} \mid * \ldots * \mid S_{is} \mid : \forall \varphi_i \in \phi\} \tag{4.4}$$

Además, $Fals(\varphi_i)$ organiza valores lógicos para un conjunto de variables que no cambian de valor durante el proceso $Ind(\varphi_i, K)$, y entonces $\mid S_{i1} \mid + \mid S_{i2} \mid + \ldots + \mid S_{is} \mid \leq n - \mid \varphi_i \mid$.

Para simplificar el análisis de complejidad de $f(|\varphi_i|, |K|)$, supongamos que K es una r-FNC, $r \in [[1, n]]$ (todas las cláusulas de K tienen r literales). En el peor de los casos, cada cláusula $C_{i1}, C_{i2}, \ldots, C_{is}$ utilizadas durante $Ind(\varphi_i, K)$ son dependientes entre sí, y son disjuntas, esto es $Lit(C_{is}) \cap Lit(C_{is+1}) = \emptyset$, para $s = 1, \ldots, m-1$. En este caso, cada$\mid S_{is} \mid = r$. Por lo tanto, obtenemos por (4.4) que$f(|\varphi_i|, |K|) \leq \underbrace{r * \ldots * r}_{(n-\varphi_i)/r} = (r^{1/r})^{n-|\varphi_i|}$.

Este último término tiene un máximo cuando $r = 3$, y por lo tanto, $T_0(|\phi|, |K|) \in O((3^{1/3})^{(n-min\{|\varphi_i|:\varphi_i \in \phi\})} * Poly(n, m))$. Con $3^{1/3} \approx 1.443$, nuestro proceso de revisión de creencias tiene una complejidad de tiempo de $O(1{,}443^{(n-min\{|\varphi_i| \ :\varphi_i \in \phi\})} \cdot Poly(n, m))$. Y nuevamente, asumiendo el peor de los casos, $\mid \varphi_i \mid = 1$, tenemos $T_0(|\phi|, |K|) \leq (1{,}443)^{n-1}$, sin considerar factores polinomiales.

Sin embargo, esta última condición de que K esté formado por literales puros y por cláusulas disjuntas es fácil de detectar (se puede hacer en tiempo lineal sobre el tamaño de K). Para este caso, la asignación parcial de falsificación s de φ_i se puede

evaluar en K ($K' = K[s]$). Y si la cláusula $Null$ está en K', entonces se cumple que $K \models \varphi_i$, en otro caso $K \not\models \varphi_i$. Porque cualquier FNC formado por literales puros siempre es satisfactorio, y en este caso hemos encontrado una asignación haciendo verdadero a K y falso a φ_i.

En resumen, el operador $Ind(\varphi_i, K)$ se desempeña de manera eficiente. Aunque, su aplicación iterativa en cada $C_j \in K$ para generar los nuevos FNC F_s tal que $(K \wedge Fs) \models \phi$, podría conducir a un crecimiento exponencial en el número de cláusulas en F_s. Hemos demostrado que en el peor de los casos, el número de cláusulas en F_s está delimitado por $O(1{,}443^{(n-min\{|\varphi_i|:\varphi_i \in \phi\})})$, siendo n el número de variables involucradas en K y ϕ.

4.3.5. Postulados KM

Como hemos mencionado antes, el paradigma más conocido de revisión de creencias es el enfoque AGM [17]. Posteriormente, Mendelzon y Katsuno, unificaron los diferentes enfoques de revisión de creencias a la semántica, y reformularon los postulados de AGM, que fueron llamados postulados de KM [39]. Posteriormente, Darwiche y Pearl propusieron la revisión iterada, donde su propuesta establece una representación basada en supuestos del modelo.

En la revisión de creencias por Fermé [29], se resumen los primeros veinticinco años de esta teoría. Los temas cubren caracterizaciones equivalentes de las operaciones de AGM, representaciones extendidas de los estados de creencias, operadores de cambio no incluidos en el marco original, cambio iterado, aplicaciones del modelo, sus conexiones con otros marcos formales, computabilidad de las operaciones de AGM y crítica del modelo.

Un teorema probado en [69] desactiva en parte el conflicto entre la revisión de AGM y la idea de una elección directa entre los resultados potenciales. Los autores demostraron que la transitividad relacional de la revisión de AGM se puede reconstruir a través de una función de transición relacional que selecciona directamente entre los resultados potenciales. Este teorema da credibilidad al enfoque de AGM, pero también al modelo general (revisión descriptiva), del cual, la revisión de AGM ahora ha demostrado ser un caso especial.

Dada la relevancia de los postulados AGM y KM en el estado del arte para la

revisión de creencias, es natural analizar cuáles de esos postulados son válidos para cualquier nueva propuesta de operadores de revisión de creencias. A continuación, presentamos el análisis de los postulados sostenidos por nuestro operador de revisión de creencias $K' = K \circ \phi = K \wedge Ind(\phi, K)$. Consideramos los tres primeros postulados propuestos por Katsuno, ya que estos postulados son más adecuados para nuestra propuesta que los tres primeros postulados de AGM. Posteriormente, consideramos los postulados originales (R4), (R5) y (R6) de AGM, que reescribimos como (U4), (U5) y (U6), ya que muestran que nuestra propuesta de revisión puede ser logrado con un cambio mínimo.

- **(U1)** $K \circ \phi \models \phi$.
- **(U2)** Si K implica ϕ entonces $K \circ \phi \equiv K$.
- **(U3)** Si ambos K y ϕ son satisfactibles, entonces $K \circ \phi$ es también satisfactible.
- **(U4)** Si $K_1 \equiv K_2$ y $\phi_1 \equiv \phi_2$, entonces $K_1 \circ \phi_1 \equiv K_2 \circ \phi_2$.
- **(U5)** $(K \circ \gamma) \wedge \phi \models K \circ (\gamma \wedge \phi)$.
- **(U6)** Si $(K \circ \phi) \wedge \gamma$ es satisfactible, entonces $K \circ (\phi \wedge \gamma) \models (K \circ \phi) \wedge \gamma$.

El teorema 4.1 muestra que nuestro operador de revisión de creencias sostiene el postulado **(U1)**.

Si K implica ϕ entonces $Fals(\phi) - Fals(K) = \emptyset$. En este caso, el operador de independencia no agrega ninguna cláusula a K, y luego $K \circ \phi = K$. Este postulado es más débil que el postulado AGM original (R2): Si $K \wedge \phi$ es satisfactible, entonces $K \circ \phi = K \wedge \phi$. Pero en nuestro caso, **(U2)** garantiza agregar a la FNC K solo las restricciones necesarias S, tales que $K \wedge S \models \phi$. Cuando S no es necesario (porque $K \models \phi$), entonces no se agrega nada a K, manteniendo la propiedad de hacer cambios mínimos en la KB original.

El postulado **(U3)** se cumple cuando $K \circ \phi$ es satisfactoria. Aunque podría darse el caso de que $Fals(K) \cup Fals(Ind(\phi, K))$ cubriera todo el espacio de asignaciones (2^n). En este caso, $(K \circ \phi)$ se redefine para mantener (U3). En este último caso, $(K \circ \phi)$ se redefine como $((K \wedge Ind(\phi, K)) - C_j)$, seleccionando una cláusula $C_j \in K$ con información mínima. Tenga en cuenta que C_j tiene información mínima cuando

$|C_j|$ es máximo en el conjunto de cláusulas de K, ya que tiene un número mínimo de modelos en 2^n. Esto nos lleva a realizar un proceso iterativo de eliminación de cláusulas de K hasta obtener una fórmula satisfactoria para $(K \circ \phi)$. La revisión de la satisfacibilidad de $(K \circ \phi)$ implica un procedimiento adicional para comprobar la satisfacibilidad en formas normales conjuntivas, que es un problema NP completo clásico.

El postulado **(U4)** se satisface porque nuestro operador de revisión es cerrado sobre formas conjuntivas, y el operador de independencia tiene $Ind(\phi_1, K) \equiv Ind(\phi_2, K)$ si $\phi_1 \equiv \phi_2$.

Sea $S_2 = K \circ (\gamma \wedge \phi)$. Entonces, $Fals(S_2) = Fals(\gamma \wedge \phi) - Fals(K) = (Fals(\gamma) \cup Fals(\phi)) - Fals(K)$. Entonces, $Fals(S_2) = (Fals(\gamma) - Fals(K)) \cup (Fals(\phi) - Fals(K))$. Por otra parte, $(Fals(\phi) - Fals(K)) \subseteq Fals(\phi)$, por lo tanto, $(Fals(S_2) \subseteq (Fals(\gamma) - Fals(K)) \cup Fals(\phi)$. Sea $S_1 = (K \circ \gamma)$. Entonces $Fals(S_1) = Fals(\gamma) - Fals(K)$. En consecuencia, $Fals(S_2) \subseteq Fals(S_1) \cup Fals(\phi) = Fals(S_1 \wedge \phi)$, pero esto sucede si y solo si $(S_1 \wedge \phi) \models S_2$, lo que implica que $(K \circ \gamma) \wedge \phi \models K \circ (\gamma \wedge \phi)$, y por lo tanto **(U5)** se cumple.

Si $(K \circ \phi) \wedge \gamma$ es satisfactible, entonces $K \circ (\phi \wedge \gamma) \models (K \circ \phi) \wedge \gamma$. Como $Fals((K \circ \phi) \wedge \gamma) = Fals(K \wedge Ind(\phi, K) \wedge \gamma)$, pero (γ) sería igual a $Ind(\gamma, K)$ si y solo si γ ya es independiente con cada cláusula de K, y en este caso, $Fals(K \wedge Ind(\phi, K) \wedge \gamma) = Fals(K \wedge Ind(\phi, K) \wedge Ind(\gamma, K)) = Fals(K \circ (\phi \wedge \gamma)$ y por tanto, el postulado **(U6)** se cumple.

En resumen, hemos utilizado los tres primeros postulados del modelo KM y los postulados originales (R4), (R5) y (R6) del modelo AGM para nuestra propuesta según las características del operador $Ind(\phi, K)$.

4.4. Algoritmo de revisión de creencias con DFS

En la búsqueda de reducir la cantidad requerida de recursos computacionales en el proceso de revisión de creencias, una forma es utilizando árboles y sus algoritmos desarrollados. Por ejemplo al trabajar con grafos si los podemos transformar a un árbol, podemos reducir el numero de operaciones [75]. Modificamos el algoritmo de revisión descrito en la sección anterior para proponer un algoritmo que utiliza la

búsqueda en profundidad (DFS o *Depth First Search*, en inglés). El Algoritmo 4.5 de la sección anterior se basa en una tabla de $|K|$ x $|\phi|$ que es ajustada dinámicamente en función del número de nuevas cláusulas a generar. De esta forma, la tabla se escanea fila a fila. Mientras el nuevo algoritmo revisa los nodos de un árbol en profundidad, pero con un valor de profundidad limitado $|k|$, esto evita que se expanda la profundidad del árbol de forma indefinida, además, se evita la necesidad de guardar todos los nodos expandidos.

Algoritmo 5 Función $Pre(F, K, S)$

para todo $C_i \in K$ **hacer**
  si $(Dif(F, C_i) = 0)$ **entonces**
    {Elimina de K si es una cláusula subsumida}
    $K \leftarrow K - C_i$
    Return K
  fin si
  si $(Ind(F, C_i) > \emptyset)$ **entonces**
    {Elimina de K una cláusula independiente }
    $S \leftarrow S + C_i$
    $K \leftarrow K - C_i$
  fin si
fin para
retornar K

$Pre(F, K, S)$ realiza un preprocesado de la base de conocimiento K respecto a cada una de las cláusulas de la nueva información ϕ con el fin de eliminar cláusulas subsumidas y cláusulas independientes. El Ejemplo 4.7 muestra la aplicación del preproceso de la base de conocimiento K.

Ejemplo 4.7 *Pre-procesamiento de la KB K.*
*Sea $K = \{C_1, C_2, C_3, C_4, C_5, C_6\}$ = {(10*0*), (*110*), (11***), (110**), (11**0), (*101*)} y ϕ ={(11**1)}, cuando se realiza el llamado a la función $K = Pre(\phi_i, K)$, obtenemos: $Ind(\phi, C_1) = true$, $Dif(\phi, C_3) = 0$. En este caso, C_3 subsume a ϕ, y no habría que hacer más trabajo porque $K \models \phi$.*

Ademas del algoritmo (6) de revisión de creencias con DFS y de los subprocesos, se utilizan los siguientes procedimientos:

- $DFS(\phi, K, S)$**:** esta función permite generar el árbol de recorrido a lo profundo para cada cláusula en ϕ.
- $DFS\_BR(\phi, K, S)$**:** este es el procedimiento principal de revisión de creencias de la nueva información ϕ y la base de conocimiento K, retornando S que representa el conjunto de cláusulas que no se pueden inferir de K y que representan además el espacio de soluciones posibles de K dado ϕ. Este procedimiento genera un árbol de recorrido a lo profundo de forma recursiva para cada cláusula ϕ.

Algoritmo 6 Procedimiento $DFS\_BR(\phi, K, S)$

Entrada:
K: a KB, ϕ: cláusula con la nueva información
salida:
S: conjunto de cláusulas resultantes
para todo $C_j \in \phi$ **hacer**
  $Fi = get(\phi)$: primera cláusula de ϕ
  $K = Pre(Fi, K)$: pre-procesamiento
  $K = sort(K, Fi)$: ordena K respecto a las diferencias con Fi
  $S = DFS(\phi, K, S)$: llamado recursivo de búsqueda a lo profundo
fin para
si (S es *vacía*) **entonces**
  $escribir(''\phi$ se infiere de $K'')$
sino
  $escribir(S)$
fin si

La propuesta del algoritmo de revisión de creencias usando la estrategia de recorrido a lo profundo se muestra en el algoritmo 6. El Algoritmo $DFS\_BR(\phi, K, S)$ representa el procedimiento general para realizar el proceso de inferencia proposicional para la revisión de creencias utilizando la técnica de recorrido a lo profundo, en el Ejemplo 4.8 se describe paso a paso la aplicación de éste algoritmo.

El Algoritmo 6 revisa cada cláusula de la nueva información ϕ respecto a la base de conocimiento K, se aplica el preprocesamiento con el procedimiento $Pre()$,

se ordena y se realiza un llamado al procesamiento recursivo $DFS()$ descrito en el algoritmo 7.

Algoritmo 7 Función $DFS(\phi, K, S)$

si (ϕ es *vacía*) **entonces**
  retornar S
fin si
si (K es *vacía*) **entonces**
  $S \leftarrow S + Pop(\phi)$ {se actualiza el conjunto S }
fin si
$Fi \leftarrow Pop(\phi)$
$C \leftarrow Pop(K)$
si ($Ind(Fi, Ci) \neq \emptyset$) **entonces**
  $Push(\phi, Fi)$ {se agrega Fi a ϕ por ser independiente}
sino
  $K\_dif \leftarrow Dif(Fi, C)$ {se obtiene la diferencia entre literales }
  si ($K\_dif > 0$) **entonces**
    $K \leftarrow K - C_i$
    $Gen(\phi, Fi, C, K\_dif)$ {se generan nuevas cláusulas }
  fin si
fin si
$K \leftarrow K - C$
$S = DFS(\phi, K, S)$ { llamado recursivo }

Ejemplo 4.8 *Sea* $K = \{C_1, C_2, C_3, C_4\}$ = {*(11****), (*1**1*), (1****1), (0*00**)*} *y* ϕ ={*(***0**)*}, *con* $S = []$. *En este caso* ϕ *no se puede subsumir y* K *ya esta ordenada. Cuando se realiza un llamado a la función* $DFS(\phi, K, S)$, *se genera un árbol de deducción.*

En la Figura 4.4 se indica el proceso inicial del recorrido a lo profundo para el ejemplo 4.8. Al aplicar la función $DFS(\phi, K, S)$, dada la cláusula inicial (***0***) se generan dos sub ramas debido a su diferencia con la cláusula C_1=(11****), $Dif(\phi, C_1) = 2$, posteriormente la cláusula de la izquierda (0**0**) se compara con C_2= (*1**1*) y genera a su vez dos sub ramas que están indicadas en el árbol como (00*0** y (01*00*).

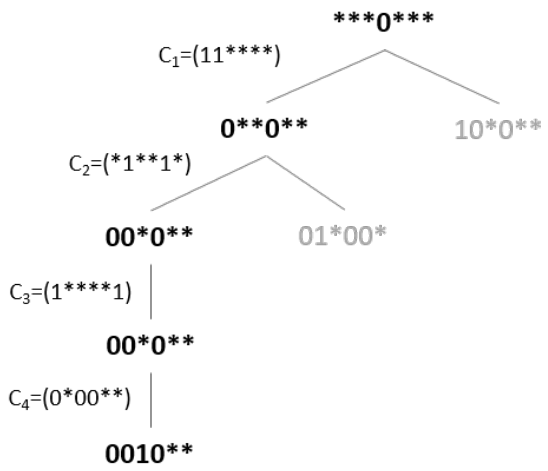


Figura 4.4: Fase inicial del algoritmo 6 $DFS(\phi, K, S)$

Se procesa la rama izquierda (00*0**) comparándola con C_3=(1***1) y genera independencia por lo que no se generan mas ramas en el árbol. Continua el proceso comparando la cláusula (00*0**) con C_4=(0*00**) y se genera la cláusula (0010**). Aquí termina el proceso en profundidad para la parte más a la izquierda, la cláusula resultante se almacena en el conjunto de salida $S = \{0010 * *\}$.

El proceso recursivo regresa el control a la cláusula padre (00*0**) y como aquí no hay más cláusulas que procesar, se regresa el control a la cláusula (0**0**).

En la Figura 4.5 se indica el proceso generado por el recorrido a lo profundo aplicado a la rama central del árbol. Al comparar la cláusula (01*00*) con C_3=(1****1) es independiente por lo que no se generan cláusulas extras y continua la evaluación ahora con C_4=(0*00**) generando así la cláusula (01100*) que se almacena en el conjunto S, por lo que ahora S={(0010**), (01100*)}.

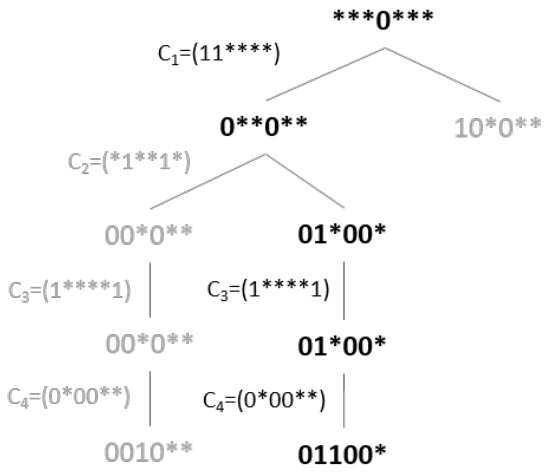


Figura 4.5: sub-árbol central generado con el algoritmo $DFS(\phi, K, S)$

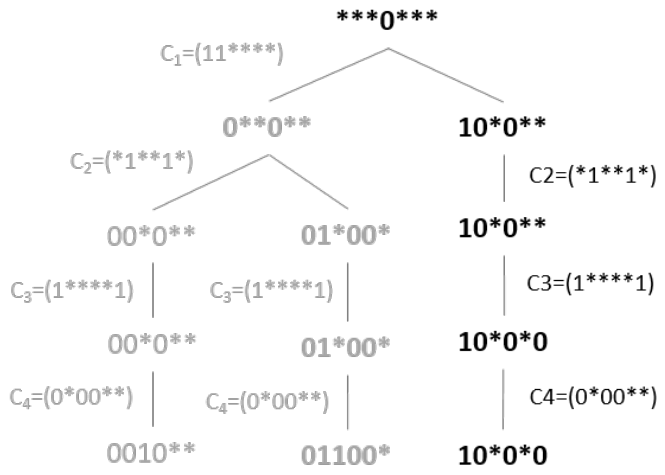


Figura 4.6: Procesamiento final de la función $DFS(\phi, K, S)$

Finalmente en la Figura 4.6 se muestra el procesamiento $DFS(\phi, K, S)$ de la última rama generada, donde al evaluar con C_2=(*1**1*) y C_3=(1****1) resulta que éstas cláusulas son independientes con (10*0**) por lo que no se generan nuevas ramas. El proceso culmina con C_4=(0*00**) dando como resultado la cláusula (10*0*0) que se agrega al conjunto S. De esta manera el conjunto S estará formado por las ramas del árbol deductivo, donde S={(0010**), (01100*), (10*0*0}.

4.4.1. Aplicación del algoritmo DFS

En el Ejemplo 4.9 se muestra la aplicación del algoritmo 7 $DFS\_BR(\phi, K, S)$.

Ejemplo 4.9 *Procesamiento del algoritmo principal. Sea* $K = \{(r \vee s \vee \neg u \vee \neg v) \wedge (r \vee \neg s \vee \neg u \vee v \vee \neg w) \wedge (\neg q \vee \neg s \vee \neg t \vee u \vee \neg w) \wedge (\neg q \vee \neg r \vee \neg s \vee \neg w) \wedge (\neg q \vee r \vee t \vee u \vee \neg w) \wedge (p \vee q \vee r \vee w)\}$ *y* $\phi = \{(\neg q \vee \neg s \vee \neg v \vee \neg w) \wedge (\neg w) \wedge (r \vee s \vee \neg u \vee \neg v \vee w)\}$. *Representando las fórmulas utilizando patrones falsificantes:*
$K = \{C_1, C_2, C_3, C_4, C_5, C_6\}$= *{(**00*11*), (**01*101), (*1*110*1), (*111***1), (*10*00*1), (000****0) }* *y* ϕ =*{ (**00*110), (*******1), (*1*1**11)}*.

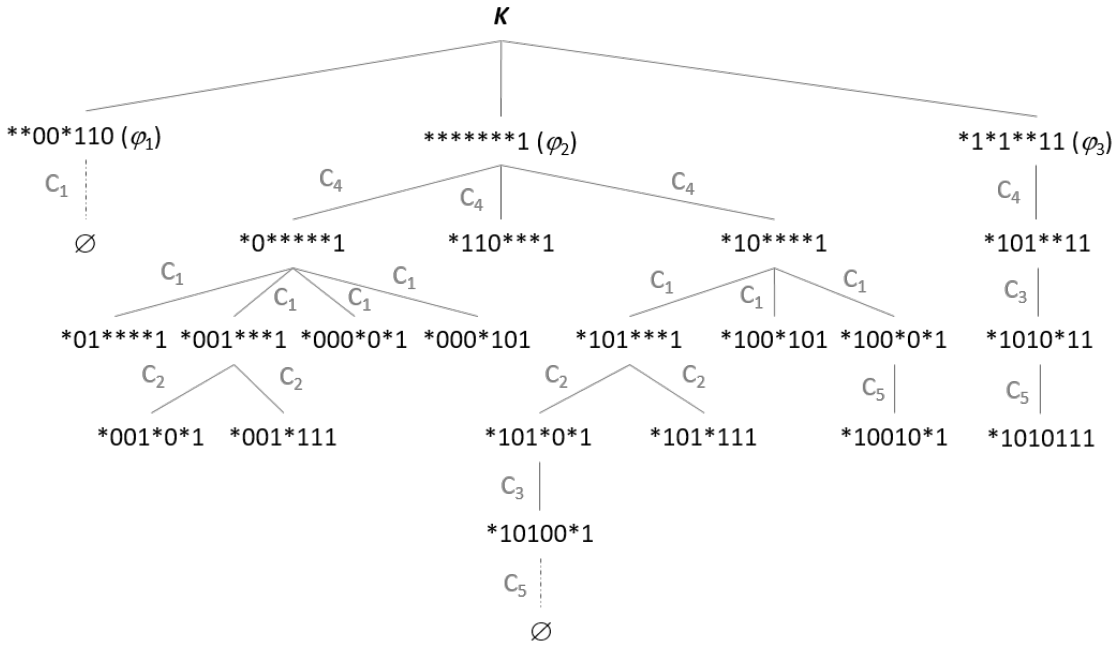


Figura 4.7: Generación del árbol de recorrido a lo profundo por el algoritmo $DFS\_BR()$

Se inicia el procesamiento recursivo generando las ramas centrales (ver Figura 4.7) , en cada arista se indica la cláusula aplicada. La línea punteada indica que la cláusula fue subsumida.

A continuación se describe paso a paso la aplicación del Algoritmo 6:

- El algoritmo inicia con un pre-porcesamiento (aplicando la función $Prep()$) de la base de conocimiento K y la cláusula φ_1. En este paso se determina que la cláusula φ_1 es una cláusula subsumida por C_1 debido a que su diferencia es cero.

- Se reinicia el proceso ahora con la cláusula φ_2, al realizar el pre proceso se quita la cláusula C_6 por ser independiente con φ_2.

- A continuación se ordena la base de conocimiento K, de acuerdo a las diferencia $(Dif())$ con la cláusula φ_2, así el orden es K=$\{C_4, C_1, C_2, C_3, C_4, C_5\}$.

- La base de conocimiento reducida K es ordenada con respecto a φ_2, resultando K=$\{C_4, C_3, C_5\}$.

- Continúa el procedimiento con la última cláusula φ_3 generando la rama más a la derecha donde al realizar el pre proceso se quitaron las cláusulas C_1, C_2 y C_6 por ser cláusulas independientes con φ_3.

- El procesamiento termina cuando se han evaluados todas las cláusulas de ϕ y las hojas del árbol representan el conjunto final de cláusulas del conjunto S. Para este ejemplo tenemos que $(K \wedge S)$=$\{$(*01****1), (*001*0*1), (*001*111), (*000*0*1), (*000*101), (*110***1), (*101*111), (*100*101), (*10010*1), (*10 10111) $\}$. El árbol resultante se muestra en la Figura 4.7.

En la Tabla 4.3 se muestran las operaciones aplicadas en el procesamiento del algoritmo $DFS\_BR()$. La base de conocimiento K está formada por 6 cláusulas mientras que la nueva información ϕ contienen 3 cláusulas. Al inicio del procedimiento podemos notar que la cláusula φ_1 es subsumida por C_1. Cuando consideramos la segunda cláusula φ_2, la cláusula C_6 es eliminada de la base K debido a que es independiente con φ_2. Entonces, la base K es ordenada y se aplica la función $DFS()$. La función $Ind()$ indica que las cláusulas son independientes por lo que no se generan más sub árboles en la búsqueda a lo profundo.

Finalmente, cuando la cláusula φ_3 es procesada, las cláusulas C_1, C_2 y C_6 son eliminadas por ser independientes con φ_3, entonces, las cláusulas restantes de K son ordenadas para aplicar nuevamente la función $DFS()$.

Tabla 4.3: Operaciones procesadas por el algoritmo $DFS\_BR()$

	Base de conocimiento original K					
	C_1	C_2	C_3	C_4	C_5	C_6
ϕ_1	**00*11*	**01*101	*1*110*1	*111****	*10*00*1	000****1
**00*110	$\emptyset$	-	-	-	-	-
			Ordena k			
	C_4	C_1	C_2	C_3	C_5	C_6
ϕ_2	*111****	**00*11*	**01*101	*1*110*1	*10*00*1	000****1
*******1	*0*****1	***01****1**	$ind()$	$ind()$	$ind()$	-
		*001***1	***001*0*1**	$ind()$	$ind()$	-
			***001*111**	$ind()$	$ind()$	-
		***000*0*1**	$ind()$	$ind()$	$ind()$	-
		***000*101**	$ind()$	$ind()$	$ind()$	-
	*10****1	*101***1	*101*0*1	*10100*1	$\emptyset$	-
			***101*111**	$ind()$	$ind()$	-
		*100*0*1	$ind()$	$ind()$	***10010*1**	-
		***100*101**	$ind()$	$ind()$	$ind()$	-
	1101**	$ind()$	$ind()$	$ind()$	$ind()$	-
			Ordena k			
	C_4	C_3	C_5	C_1	C_2	C_6
ϕ_3	*111****	*1*110*1	*10*00*1	**00*11*	**01*101	000****1
*1*1**11	*101**11	*1010*11	***1010111**	-	-	-

4.4.2. Consistencia de la base de conocimiento

Mantener la consistencia de la base de conocimiento en el proceso de revisión de creencias es un elemento que se debe considerar al agregar nueva información. En este trabajo se presenta un método que permite determinar cuando la base de conocimiento deja de ser consistente al añadir un conjunto de cláusulas que representan la nueva información [73].

Recordemos que dada K una base de conocimiento y dada ϕ la nueva información, ambas expresadas como un conjunto de cláusulas en FNC con la condición de que $K = \bigwedge_{(j=1)}^{m} C_j$ y $\phi = \bigwedge_{(i=1)}^{k} \varphi_i$ donde cada $C_j \in K$ y cada $\varphi_i \in \phi$ son expresadas en el mismo conjunto de n variables Booleanas.

Para el proceso de revisión de la consistencia de la base de conocimiento utilizamos el proceso base de la revisión de creencias, y aplicamos una revisión sobre el conjunto S:

- Se inicia aplicando el algoritmo $DFS\_BR()$ de inferencia proposicional. Sea KB K y ϕ la nueva información, obtenemos S como el conjunto resultante de $Ind(K, \Phi)$.

- Si $S = \emptyset$ entonces no se agrega ninguna cláusula a la base de conocimiento K, de esta manera se mantiene su consistencia.

- Si $S \neq \emptyset$ entonces, se aplica un proceso de revisión $(K * S)$ sobre el conjunto S y obtenemos S'. Utilizamos las ecuaciones 4.5 y 4.6 para este proceso.

$$x = 2^{\#asteriscos} \tag{4.5}$$

$$Si(\#Mod(K) - \sum_{i=1}^{m} \#Mod(S_i)) \tag{4.6}$$

- Si $\#Mod(S') < \#Mod(K)$, entonces, K es consistente y se aplica un proceso de expansión $K = (K + S')$.

- Si $\#Mod(S') \geq \#Mod(K)$, entonces, K no es consistente y se debe aplicar un proceso de contraer $(K \cup S')$.

Para determinar si la base de conocimiento K continúa siendo satisfactible (consistente) es necesario aplicar un proceso de revisión sobre S para obtener S' y contar el número de modelos $Mod(S')$, de acuerdo a la fórmula (4.5), que se debe restar a los modelos de K, es decir, aplicar la fórmula (4.6).

Ejemplo 4.10 *Verificar la consistencia de K, sea la KB $K = \{C_1, C_2, C_3, C_4\} = \{(\neg q \vee r \vee s \vee \neg t) \wedge (\neg p \vee \neg q \vee s) \wedge (\neg q \vee \neg s) \wedge (\neg p \vee \neg r)\}$ y sea $\phi = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4\} = \{(\neg p \vee \neg q \vee r) \wedge (q \vee \neg r) \wedge (p \vee \neg r \vee \neg s) \wedge (\neg t)\}$. Transformando cada cláusula usando cadenas falsificantes en el mismo espacio de variables p, q, r, s, t, obtenemos: K=\{(*1001), (11*0*), (*1*1*), (1*1**)\}, donde $\#SAT(K) = \#Mod(K) = 15$ y sea ϕ= \{(110**) (*01**) (0*11*) (****1)\}.*

- El proceso principal inicia aplicando una revisión de cláusulas subsumidas. En este caso no hay cláusulas subsumidas en ϕ.

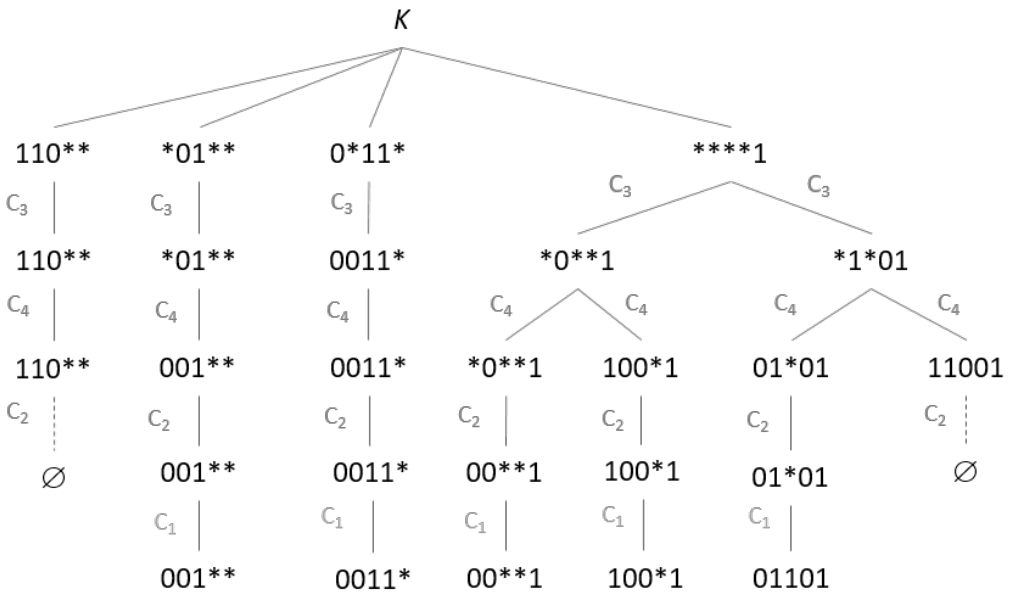


Figura 4.8: Ejcución del algoritmo de inferencia para el ejemplo 4.10

- Se ordena la base de conocimiento K de acuerdo al número de asteriscos, entonces $K = \{C_3, C_4, C_2, C_1\}$.

- Inicia el proceso recursivo comparando cada cláusula de ϕ_i contra la cláusula C_3, como se muestra en el nivel uno del árbol de la Figura 4.8. La cláusula vacía $\varnothing$ indica que se llega a una cláusula subsumida.

- Note que la cláusula ϕ_4 genera dos cláusulas más debido al concepto de diferencia.

- El recorrido a lo profundo finaliza cuando todas las cláusulas son evaluadas de ϕ y las hojas del árbol representan las cláusulas finales de S. Para este ejemplo, tenemos que $S = \{(001**), (0011*), (00**1), (100*1), (01101)\}$.

Aplicamos el proceso de revisión sobre S. Cada fila y columna esta etiquetada con las cláusulas de S, y obtenemos cláusulas independientes o cláusulas subsumidas.

Si la cláusula es subsumida significa que los modelos de dicha cláusula ya están contenidos en alguna otra cláusula de S. En cambio, si la cláusula es independiente

entonces esta cláusula debe incluirse en S' y al ser independiente restará cláusulas a la base de conocimiento K.

Tabla 4.4: Aplicando revisión sobre S

S	01101	0011*	100*1	001**	00**1	S'
01101	-	Ind	Ind	Ind	Ind	01101
0011*	-	-	Ind	Sub	-	-
100*1	-	-	-	Ind	Ind	100*1
001**	-	-	-	-	001*0	001*0
00**1	-	-	-	-	-	00**1

De esta forma después de aplicar este proceso de revisión obtenemos el conjunto S' como se muestra en la Tabla 4.4.

El total de modelos de la Tabla 4.4 representa la suma $\#Mod(S'_i) = 9$, este valor se obtiene aplicando la fórmula (4.5) sobre cada fila de acuerdo al número de asteriscos, por ejemplo $(001^{**}) = 2^2 = 4$ modelos. De esta forma aplicando la fórmula (4.6), $\#Mod(K) - \#Mod(S') = 15 - 9 = 6$ modelos, Entonces $(K \cup S')$ es consistente y se puede actualizar la base de conocimiento K de forma $K = K + S'$.

El conjunto S' de cláusulas contiene exactamente las cláusulas necesarias que permiten inferir cada $\varphi_i \in \phi$ de KB K. El conjunto de cláusulas S' representa el mínimo conjunto de cláusulas que permiten cubrir el espacio $Fals(\varphi_i)$ - $Fals(C_j)$ con $C_j \in K$.

Capítulo 5

Diseño del Algoritmo de conteo de modelos

Presentamos aquí una propuesta algorítmica para el conteo de modelos, que es aplicada a la base de conocimiento K, con el fin de determinar si K es satisfactible [74].

Consideramos para cada cláusula el mismo conjunto de n variables y formamos cadenas de ceros, unos y asteriscos (*), posteriormente aplicamos un proceso de revisión para generar cláusulas independientes entre sí, es decir cláusulas que tengan literales complementarias.

El método de conteo de modelos es iterativo tomando una a una las cláusulas de la fórmula F en FNC y que representa la base de conocimiento K, iniciamos con el valor de 2^n modelos y se le van restando modelos de acuerdo al número de asteriscos ($2^{\#asteriscos}$) de las cláusulas resultantes al aplicar el proceso de revisión. En la Figura 5.1 se muestra el proceso completo para la propuesta de conteo de modelos, para verificar la consistencia de la base de conocimiento modelada como una fórmula F. Además, se incluye el proceso para el caso de que F sea inconsistente (contracción de la base de conocimiento).

- Si una fórmula F es satisfactible entonces la base de conocimiento es consistente, es decir, tiene por lo menos un modelo, en caso contrario, la base de conocimiento será inconsistente.

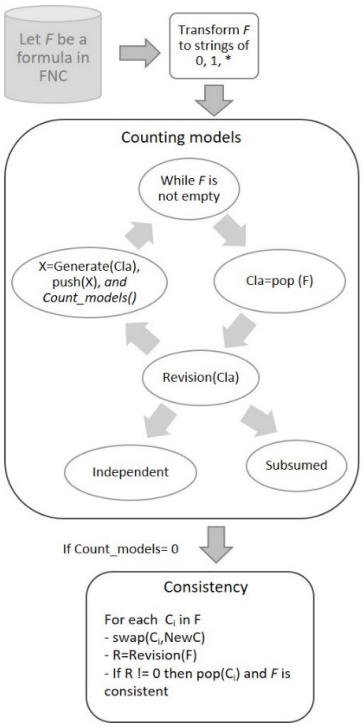


Figura 5.1: Proceso completo para el conteo de modelos

- El problema $\#SAT(F)$ representa el número de modelos que hacen satisfactible a la fórmula F.

Cuando transformamos una cláusula $C \in F$ en base al mismo conjunto de n variables, tenemos que el número de modelos (Mod) satisfactibles está dado por $Mod(F) = 2^n - 2^{\#asteriscos(C)}$, donde $\#Asteriscos(C)$ es el número de asteriscos de la cláusula C.

Por ejemplo sea $F = \{(1***0)\}$ con $n = 5$ variables, por lo tanto $asteriscos(C) = 3$, entonces $Mod(F) = 2^5 - 2^3 = 32 - 8 = 24$ modelos.

En el proceso general de la propuesta algorítmica para la revisión de la consistencia de una base de conocimiento utilizamos tres operaciones básicas descritas en 4.3.

La propuesta de algoritmo de revisión de una base de conocimiento se presenta en el Algoritmo 8.

Las operaciones de $Ind()$ y $Gen()$ corresponde a ciclos simples de n variables, las operaciones son excluyentes para cada cláusula. Cuando se llega al caso de cláusula

Algoritmo 8 $CuentaModelos(\varphi_i, K)$

Entrada: $F = \{f_1, f_2, ..., f_m\}$: una KB en FNC, con m cláusulas y n variables en cada cláusula
Salida: S conjunto de cláusulas equivalentes resultantes del conteo
$TotalMod$: total de modelos de la base de conocimiento
Ordenamos(F); {Se ordena en base al número de asteriscos de cada cláusula}
$TotalMod = 2^n - 2^{asteriscos(C_1)}$; { C_1 es la cláusula con más asteriscos};
$S = C_1$;
para todo $f_i \in F, i = 2$ **hasta** m **hacer**
  $C = f_i$;
  mientras (S tenga cláusulas no visitadas) **hacer**
    $in = Ind(S_i, C)$; {Verifica independencia}
    $nc = Gen(S_i, C)$; {Nuevas cláusulas}
    marcar_visitadas(S_i);
    si ($n = 1$ y cláusulas visitadas en S) **entonces**
      $S = agregar(C)$;
    sino
      si ($nc > 0$) **entonces**
        para todo $j = 0$ hasta nc **hacer**
          $F = agregar(nuevas\_clausulas_j)$
        fin para
      sino
        $S = agregar(C)$;
        break; {Cláusula subsumida}
      fin si
    fin si
    $x = \#asteriscos(C_i)$ {# asteriscos de una cláusula}
    $TotalMod = TotalMod - 2^x$
  fin mientras
fin para

subsumida se detiene el ciclo para continuar con otra cláusula de la fórmula F.

La generación de nuevas cláusulas aumenta el proceso de revisión con las nuevas cláusulas generadas y para el caso de independencia al terminar de revisar toda la fórmula F se agrega a la salida S que representa la fórmula reducida para el conteo de modelos.

Para el caso de que el número de modelos ($TotalMod$) de F sea 0 al aplicar el algoritmo de conteo, entonces, será necesario eliminar alguna cláusula con el fin de mantener la consistencia de la fórmula y de la base de conocimiento, a este proceso se le denomina: contración de la base de conocimiento.

Para ilustrar como se revisa la consistencia de la base de conocimiento, consideremos el siguiente ejemplo.

Ejemplo 5.1 *Sea* $F = \{(q \vee \neg r \vee \neg s) \wedge (\neg p \vee \neg r) \wedge (\neg q) \vee (q \vee s) \wedge (\neg p \vee \neg q \vee r \vee s)\}$ *una fórmula en FNC con* $m = 5$ *cláusulas y* $n = 4$ *variables. Transformamos cada cláusula con el patrón de ceros, unos y asteriscos, obtenemos* $F = \{$*(*100), (0*0*), (*0**), (*1*1), (0011)*$\}$.

De acuerdo al algoritmo primero se ordena F considerando el número de asteriscos que tiene cada cláusula, entonces, $F = \{$(*0**), (0*0*), (*1*1), (*100), (0011)$\}$. El considerar la cláusula con más asteriscos como inicial nos permite reducir el número de operaciones a realizar.

Se inicia el procesamiento del Algoritmo 8, de conteo de modelos con:
$S = C_1 = \{(*0**)\}$;
$totalMod = 16–2^{\#asteriscos(C_1)} = 16 - 8 = 8$;
$TotalMod$ es la variable que inicia con el máximo de modelos posibles de F ($2^n = 2^4 = 16$ modelos), a los cuales se les resta los modelos de la primera cláusula $2^3 = 8$.

La tabla 5.1 describe el proceso iterativo de conteo de modelos de F. Al procesar la cláusula C_2, se aplica la función $Gen()$ que genera la cláusula (010*), esta cláusula se agrega al conjunto S. Cuando procesamos la cláusula C_3, se compara con cada cláusula del conjunto S y se generan dos cláusulas (11*1) y (0111), la primera resta dos modelos por tener un asterisco y la segunda resta 1 modelo por tener 0 asteriscos.

El proceso continúa hasta procesar la cláusula C_5 que es subsumida con la primera cláusula de S por lo que ya no es necesario seguir comparando con las demás cláusulas y tampoco resta modelos a $totalMod$, por lo que la base de conocimiento F

tiene $totalMod = 2$ modelos, de esta forma, la base de conocimiento es consistente. S={(*0**), (010*), (11*1), (0111), (1100)} es una base de conocimiento equivalente con las cláusulas necesarias para el conteo de modelos y representa el conocimiento necesario para inferir a la misma base de conocimiento F.

Tabla 5.1: Conteo de modelos del Ejemplo 5.1

i	C_i	S					$totalMod$
		*0**					
2	0*0*	010*					8-2=6
		*0**	010*				
3	*1*1	*1*1	11*1				
			0111				6-3=3
		*0**	010*	11*1	0111		
4	*100	*100	1100	1100	1100		3-1=2
		*0**	010*	11*1	0111	1100	
5	0011	$Sub()$					2-0=2

Ejemplo 5.2 *Sea* $F = \{C_1, C_2, C_3, C_4\} = \{(\neg p \vee q \vee \neg r) \wedge (r) \wedge (p \vee \neg r) \vee (\neg q \vee \neg r)\}$ *una fórmula en FNC con* $m = 4$ *cláusulas y* $n = 3$ *variables. Transformamos cada cláusula con el patrón de ceros, unos y asteriscos, obtenemos* F = {*(010), (**1), (1*0), (**0)*}.

Ordenamos $F = \{(**1), (1*0), (0*0), (010)\}$, iniciamos la aplicación del algoritmo con:
$S = C_1$={(**1)};
$totalMod = 8–2^{\#asteriscos(C_1)} = 8 - 4 = 4$;
La Tabla 5.2 muestra el proceso del conteo de modelos para el Ejemplo 5.2, en este caso la fórmula F es insatisfactible debido a que el numero de modelos es 0 ($totalMod = 0$). En este caso se debe aplicar un proceso de contracción.

5.1. Contracción de la base de conocimiento

La operación de revisión ($K \circ \phi$), con K y ϕ dos FNC's, hace que el número de cláusulas en K se incremente, en tanto que el número de modelos de ($K \cup \phi$) decrece. Sin embargo, las operaciones $Ind()$, $Sub()$ y $Gen()$ a través de los patrones de 0,1 y * nos permiten conocer el número de modelos que se van perdiendo al agregar nueva información, es decir, cuando el número de modelos llega a ser cero a medida que se van agregando nuevas cláusulas a K.

Tabla 5.2: Conteo de modelos del Ejemplo 5.2

i	C_i		S		$totalMod$
		**1			
2	1*0	1*0			4-2=2
		**1	1*0		
3	0*0	0*0	0*0		2-2=0
		**1	1*0	0*0	
4	010	010	010	$sub()$	0-0=0

Cuando el número de modelos de la base de conocimiento es cero, entonces, es necesario aplicar un proceso de contracción ($K - \phi$) ($K - C_i$), con C_i en K, de acuerdo a la teoría de revisión de creencias, se debe eliminar el conocimiento más antiguo.

La estrategia de contracción propuesta consiste en aplicar el proceso central de revisión con las operaciones $Ind(), Gen()$ y $Sub()$, bajo las siguientes consideraciones:

- Utilizamos el principio de cambio mínimo y privilegiando la nueva información (ϕ) respecto a la anterior (K) según la teoría AGM.

- Consideramos la nueva información ϕ, como la última cláusula que llego a la base de conocimiento.

- El proceso de contracción se aplica sobre la base de conocimiento original K (este caso sobre F).

- Si K y ϕ son cláusulas expresadas bajo un mismo conjunto de n variables Booleanas, entonces el número máximo de modelos es 2^n, de esta forma cada cláusula φ_i al ser independiente de K restará los modelos de acuerdo a la Ecuación (4.5).

La estrategia de contracción consiste en aplicar un proceso de revisión intercambiando la nueva información ϕ por alguna cláusula de la base de conocimiento K. Obtenemos el número de modelos que se pueden recuperar de la base de conocimiento al agregar ϕ.

Consideremos el Ejemplo 5.3, primero realizamos el conteo de modelos (ver Tabla 5.3 para verificar que $Mod(K) = 0$.

Ejemplo 5.3 *Consideremos la base de conocimiento* $K = \{(q \vee \neg r \vee \neg s) \wedge (\neg p \vee \neg r) \wedge (\neg q) \wedge (q \vee s) \wedge (\neg p \vee \neg q \vee r \vee s)\}$ = *{(*100), (0*0*), (*0**), (*1*1), (0011), (***0)}, al aplicar el algoritmo de conteo, Algoritmo 8, se obtiene que el número de modelos de* $K = totalMod = 0$ *(ver Tabla 5.3).Por lo que ahora debemos aplicar el proceso de contracción.*

Tabla 5.3: Conteo de modelos del Ejemplo 5.3

i	C_i			S			$totalMod$
		***0					
2	*0**	*0*1					8-4=4
		***0	*0*1				
3	0*0*	0*01	0101				4-1=3
		***0	*0*1	0101			
4	*1*1	*1*1	*1*1	11*0			3-2=1
				0111			1-1=0
		***0	*0*1	0101	11*1	0011	
5	*100	$Sub()$					0-0=0
		***0	*0*1	0101	11*1	0011	
6	0011	0011	$Sub()$				0-0=0

Aplicando un proceso de contracción sobre K, en el Ejemplo 5.3, donde $F = \{(q \vee \neg r \vee \neg s) \wedge (\neg p \vee \neg r) \wedge (\neg q) \wedge (q \vee s) \wedge (\neg p \vee \neg q \vee r \vee s)\} = \{$(*100), (0*0*), (*0**), (*1*1), (0011), (***0)$\}$, para elegir la cláusula candidata a ser eliminada de la base de conocimiento, intercambiamos la última cláusula (nueva información ϕ=(***0)) por cada una de las cláusulas C_i de K. Este proceso se muestra en la Tabla 5.4.

Al intercambiar la cláusula C_1 contra la nueva información ϕ observamos que se recuperan 0 modelos, de igual forma sucede con la cláusula C_2. De esta forma notamos que la cláusula $C_3 = (*0**)$ al ser intercambiada con ϕ en la base de conocimiento se recuperan 2 modelos, mientras que la cláusula $C_4 = (*1*1)$ recupera 3 modelos.

Considerando eliminar la cláusula C_3. La base de conocimiento se actualiza $K = \{(*100)(0*0*)(*1*1)(0011)(***0)\} = \{(q \vee \neg r \vee \neg s) \wedge (\neg p \vee \neg r) \wedge (q \vee s) \wedge (\neg p \vee \neg q \vee r \vee s) \vee (\neg s)\}$ contando con 2 modelos.

Tabla 5.4: Contracción de la base de conocimiento

Cláusula a eliminar	K					Modelos recuperados
	***0	0*0*	*0**	*1*1	0011	
*100	$Sub()$					0
	*100	***0	*0**	*1*1	0011	
0*0*	000*	0001	$Sub()$			0
	*100	0*0*	***0	*1*1	0011	
*0**	*0**	10**	10*1	10*1	10*1	2
		001*	0011	0011	$Sub()$	0
	*100	0*0*	*0**	***0	0011	
*1*1	11*1	11*1	11*1	11*1	11*1	2
		0111	0111	0111	0111	1
	*100	0*0*	*0**	*1*1	***0	
0011	0011	0011	$Sub()$			0

En el proceso de conteo de modelos, si una cláusula C_k de la base K es una cáusula subsumida ($Sub()$), entonces, la cláusula C_k no es candidata a ser eliminada.

Una cláusula subsumida no modifica el número de modelos de K. Por ejemplo, en la Tabla 5.3 las cláusulas (*100) y (0011) son subsumidas, por lo tanto, al realizar el proceso de contracción no recuperarán ningún modelo, por lo que el número de modelos de K no cambia.

Ejemplo 5.4 *En el área de razonamiento automático se modelan problemas con lógica proposicional, por ejemplo, dado el siguiente conjunto de proposiciones que representan una base de conocimiento de un asistente virtual A para la compra de pañales en línea, supongamos que parte de las instrucciones son: p: comprar pañales, q: pañales de marca, r: marca blanca, s: pañales de tela, t: paquete individual, u: cliente desperdicio cero, w: empaque biodegradable.*

Formamos el conjunto de instrucciones del asistente virtual A del Ejemplo 5.4:

- Comprar pañales (p)
- No hay pañales ó no hay pañales de marca ó hay marca blanca ($\neg p \vee \neg q \vee r$)
- No hay empaque biodegradable ó hay pañales de marca ($\neg w \vee q$)
- No hay paquete individual o hay pañales de tela ($\neg t \vee s$)
- No hay pañales de tela o cliente es del tipo desperdicio cero ($\neg s \vee u$)
- No es un cliente desperdicio cero o empaque biodegradable ($\neg u \vee w$)
- Hay paquete individual (t)
- No hay pañales de marca o hay pañales de tela ($\neg q \vee s$)
- No hay empaque biodegradable ($\neg w$)

Trasformando A en su FNC obtenemos $A = \{(p) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg w \vee q) \wedge (\neg t \vee s) \wedge (\neg s \vee u) \wedge (\neg u \vee w) \wedge (t) \wedge (\neg q \vee s) \wedge (\neg w)\} = \{(1******), (001****), (*1****0), (***10**), (***0*1*), (*****01), (****1**), (*0*1***), (******0)\}$.

La base de conocimiento A es inconsistente ya que no tiene modelos, el conteo se muestra en la Tabla 5.5.

Tabla 5.5: Contracción de la base de conocimiento

C_i	S						$totalMod$
	1******						
****1**	0***1**						64-32=32
	1******	0***1**					
******0	0*****0	0***0*0					32-16=16
	1******	0***1**	0***0*0				
*1****0	01****0	01**0*0	$Sub()$				16-0=16
	1******	0***1**	0***0*0				
***10**	0**10**	0**10**	0**10*1				16-8=8
	1******	0***1**	0***0*0	0**10*1			
***0*1*	0**0*1*	0**001*	0**0011	0**0011			8-4=4
	1******	0***1**	0***0*0	0**10*1	0**0011		
*****01	0****01	0***001	0***001	0**0001	0**0001		4-4=0
	1******	0***1**	0***0*0	0**10*1	0**0011	0**0001	
*0*1***	00*1***	00*10**	00*10*1	$Sub()$			0-0=0
	1******	0***1**	0***0*0	0**10*1	0**0011	0**0001	
001****	001****	001*0**	001*0*1	00100*1	0010001	$Sub()$	0-0=0

El conteo de modelos de A se inicia con la cláusula que tienen más asteriscos (1******) ya que es la que resta más modelos al máximo de modelos de A, en este caso se tienen $n = 7$ variables, por lo que el máximo de modelos es $2^7 = 128$. De esta forma la cláusula (1******) le resta $2^6 = 64$ modelos, por lo tanto, el algoritmo de conteo inicia con $totalMod = 128 - 64 = 64$ modelos.

Ahora aplicamos el proceso de contracción de la base de conocimiento A para determinar la cláusula candidata a eliminar de A. El proceso de revisión se muestra en la Tabla 5.6. La cláusula (******0) es considerada la nueva información (ϕ) y en la tabla se muestra cómo se va intercambiando por cada una de las cláusulas para determinar la cláusula candidata a eliminarse. La cláusula que nos permite restaurar el mayor número de modelos es (***10**), entonces $\#Mod(A) = 2$.

5.1.1. Análisis del algoritmo

La complejidad computacional de la revisión de la base de conocimiento utilizando lógica proposicional como medio de representación se encuentra en el segundo nivel de la jerarquía polinomial [22]. En nuestro caso, usamos las representaciones en FNC tanto de la base de conocimiento K como de la nueva información ϕ. El proceso de revisión de creencias $(K \circ \phi)$ se basa en resolver primero $K \models \phi$, que es un problema Co-NP sobre FNC's.

Una forma práctica de resolver este tipo de problemas difíciles, es realizar si es posible, las tareas más difíciles offline, mientras que las tareas más frecuentes se realizan en online y que, además, puedan tener un costo computacional menor.

Por ejemplo, al transformar K en un conjunto lógicamente equivalente K' pero formado sólo por cláusulas independientes, permite que el proceso de inferencia $K' \models \varphi_i$, siendo φ_i una sola cláusula, se haga en tiempo $O(n * 2^r)$, con r el número de asteriscos en φ_i. Por lo que entonces $K' \models \phi$ se realiza en tiempo $O(n * |\phi| * 2^{ri})$, con ri siendo el número máximo de asteriscos para alguna $\varphi_i \in \phi$.

Sin embargo, el mayor costo computacional del problema original es trasladado al proceso de reducción de K a K'. Esta reducción se puede realizar en tiempo $O(N * |K| * n)$, donde N será el número máximo de cláusulas independientes que se generan a partir de una cláusula $C_i \in K$. N está acotado superiormente por 2^r, con r el número de asteriscos en C_i lo que puede ser un número exponencial con respecto a n.

Aplicando la reducción de K a K' off-line, se tiene la posibilidad de trabajar online el proceso de revisión de creencias $(K' * \phi)$ de una forma más práctica que para el caso $(K * \phi)$, y con la ventaja de que al mismo tiempo se realiza el conteo de modelos que permanecerán en $(K' \cup S)$, lo que permite determinar la consistencia de la nueva base de conocimiento.

Tabla 5.6: Contracción de la base de conocimiento

C_i				A					Rec
	******0	001****	*1****0	***10**	***0*1*	*****01	****1**	*0*1***	
1******	1*****1	1*****1	1*****1	1**0**1	1**0*01	$Sub()$			
				1**11*1	1**11*1	1**1111	$Sub()$		0
	1******	******0	*1****0	***10**	***0*1*	*****01	****1**	*0*1***	
001****	001****	001***1	001***1	0010**1	0010*01	$Sub()$			
				00111*1	00111*1	0011111	$Sub()$		0
	1******	001****	******0	***10**	***0*1*	*****01	****1**	*0*1***	
*1****0	01****0	01****0	$Sub()$						0
	1******	001****	*1****0	******0	***0*1*	*****01	****1**	*0*1***	
***10**	0**10**	01*10**	01*10*1	01*10*1	01*10*1	01*1011	01*1011	01*1011	
		00010**	00010**	00010*1	00010*1	0001011	0001011	$Sub()$	2
	1******	001****	*1****0	***10**	******0	*****01	****1**	*0*1***	
***0*1*	0**0*1*	01*0*1*	01*0*11	01*0*11	01*0*11	01*0*11	01*0011	01*0011	
		0000*1*	0000*1*	0000*1*	0000*11	0000*11	0000011	0000011	3
	1******	001****	*1****0	***10**	***0*1*	******0	****1**	*0*1***	
*****01	0****01	01***01	01***01	01*0*01	01*0*01	01*0*01	01*0001	01*0001	
				01*1101	01*1101	01*1101	$Sub()$		
		000**01	000**01	0000*01	0000*01	0000*01	0000001	0000001	
				0001111	0001111	0001111	$Sub()$		3
	1******	001****	*1****0	***10**	***0*1*	*****01	******0	*0*1***	
****1**	0***1**	01**1**	01**1*1	01**1*1	01*11*1	01*1111	01*1111	01*1111	
					01*0101	$Sub()$			
		000*1**	000*1**	000*1**	00011**	000111*	0001111	$Sub()$	
						0001100	$Sub()$		
					000010*	0000100	$Sub()$		2
	1******	001****	*1****0	***10**	***0*1*	*****01	****1**	******0	
*0*1***	00*1***	0001***	0001***	00011**	00011**	000111*	$Sub()$		
						0001100	$Sub()$		0

Capítulo 6

Conclusiones

La revisión de creencias permite modelar un aspecto muy general del razonamiento humano, referente a la forma en que administramos nuestro conocimiento. En el proceso de adquisición de conocimiento, es común insertar nueva información y eliminar información errónea, o que ya no es vigente. Uno de los mecanismos que realizamos en el proceso de adquisición de conocimiento, es la aplicación de la inferencia lógica, es decir, verificar si la nueva información se puede inferir de algún conocimiento previo, en cuyo caso, ya no es necesario guardar la nueva información. El modelar el proceso de inferencia lógica de manera computacional, es uno de los retos principales en las ciencias computacionales y, en particular, en la inteligencia artificial.

Las teorías de revisión de creencias constituyen un enfoque relativamente novedoso en el área del diseño de agentes inteligentes. El utilizar operadores de revisión de creencias nos permitirá modelar problemas prácticos mediante agentes inteligentes. El estudio de la teorías de revisión de creencias se enriquece de los resultados de la lógica proposicional, lógica de predicados, lógica deóntica, entre otras líneas de investigación.

Los resultados teóricos sobre las teorías de revisión de creencias revisten interés a la hora de diseñar respuestas a cuestiones concretas de inteligencia artificial, como es en la actualización de bases de datos, o por ejemplo, en el problema de qué tipo de instrucciones es conveniente dar a un asistente virtual, así como en otras aplicaciones de la inteligencia artificial.

6.1. Aportación del trabajo de investigación

Las aportaciones de éste trabajo de investigación se centran en el desarrollo de algoritmos para el proceso general de revisión de creencias. Entre los algoritmos desarrollados en este trabajo, se tienen: algoritmo de revisión de creencias general, algoritmo de revisión de creencias con recorrido a lo profundo y el algoritmo de conteo de modelos. Se enfatiza que estos algoritmos pueden ejecutarse en un ambiente web. Los algoritmos desarrollados se sustentan de los siguientes resultados de nuestra investigación:

- Cada FNC de entrada se transforma en patrones falsificantes de 0,1 y * con el fin de facilitar las operaciones lógicas entre cláusulas. El hecho de realizar esta transformación en base a un mismo conjunto de variables nos permite conocer el número de asignaciones falsificantes en cada cláusula y, por tanto, cuantos modelos se irián restando a una base de conocimiento que se actualiza de forma dinámica.

- Se diseñó un método para la revisión de creencias entre formas normales conjuntivas. Cuando K y ϕ son FNC's, el proceso de revisión de creencias entre K y ϕ, denotado por $(K \circ \phi)$, se traduce a hacer la revisión entre cada $\varphi_i \in \phi$ y cada $C_j \in K$. Esta reducción traslada el problema general, de revisión de creencias, en resolver $(|K| * |\phi|)$ subproblemas de emparejamiento entre cadenas $(C_j \circ \varphi_i)$. Para realizar el emparejamiento $(C_j \circ \varphi_i)$, se aplica un operador lógico, llamado $Ind(\varphi_i, C_j)$.

- Se diseñó un operador lógico entre dos cláusulas , denotado por $S = Ind(\varphi_i, C_j)$, que encuentra el conjunto mínimo de cláusulas S, cuyas asignaciones de falsificación cubren el espacio de soluciones de $(Fals(\varphi_i) - Fals(C_j))$, y por lo tanto, hace que se cumpla la inferencia: $(C_j \wedge S) \models \varphi_i$.

- El resultado anterior se generaliza para construir un algoritmo de inferencia proposicional sobre formas normales conjuntivas que recibe como entrada la base de conocimiento K, la nueva información ϕ, y retorna como salida, el conjunto de cláusulas S necesarias para inferir ϕ, esto es, se construye el conjunto de cláusulas S tal que $(K \cup S) \models \phi$.

- Se muestra la solidez de la propuesta de revisión de creencias al comprobar que el operador propuesto cumple con los postulados KM.

- También se realiza la complejidad en tiempo para nuestras propuestas de revisión de creencias.

- La propuesta algorítmica establece de manera práctica un método para determinar la consistencia de una base de conocimiento K cuando se agrega nueva información ϕ, ambas fórmulas expresadas en FNC, en base a la aplicación de la inferencia proposicional $K \models \phi$. Esta propuesta representa el proceso completo de revisión de creencias utilizando las operaciones básicas del modelo AGM: Expansión, Revisión y Contracción.

- La propuesta algorítmica se puede aplicar off-line para construir una base de conocimiento K con sólo cláusulas independientes, lo que permite conocer el número de modelos que hay en K, y así trabajar online sólo el proceso de revisión de creencias $(K \circ \phi)$. Trabajar con cláusulas independientes da la ventaja de que al mismo tiempo que se ejecuta $(K \circ \phi)$ se realiza el conteo de modelos que permanecerán, y por tanto, se revisa la consistencia de la nueva base de conocimiento.

- Se contabiliza el número de modelos que hay en el conjunto $(K \cup S)$. Cuando este conjunto tiene cero modelos, entonces se aplica un proceso de contracción de alguna cláusula en K, con el fin de mantener la consistencia del conjunto $(K \cup S)$.

- Este trabajo pone de manifiesto un método para el control de la consistencia de una base de conocimiento dinámica. Por ejemplo, cuando ϕ es una cláusula unitaria $(******x)$ que involucra a una nueva variable $(n+1)$ no considerada anteriormente en K, se deben actualizar los patrones falsificantes agregando un * al final de cada cláusula de K. Esto también implica que el número de modelos por cláusula y el número de modelos de K se dupliquen, es decir se multiplican en un factor de $2^1 = 2$. Para el caso de dos nuevas variables, se agregan dos asteriscos a cada cláusula y el número de modelos se multiplican por un factor de $2^2 = 4$, y así sucesivamente para el crecimiento 2^m, donde m

serían las nuevas variables que provienen de ϕ. Para este caso, de ir agregando nuevas variables en la información ϕ, nuestro método propuesto para la revisión de la consistencia de la base de conocimiento continúa funcionando sin cambios.

- Se implementó un sistema web de revisión de creencias de gran utilidad para realizar pruebas con diferentes instancias de la base de conocimiento y la nueva información, incluyendo un módulo de generación aleatoria de cláusulas para formar la base de conocimiento y la nueva información.

- Se implementó un sistema web para contar el número de modelos de la base de conocimiento. Este sistema permite realizar pruebas para verificar si dicha base de conocimiento es consistente o inconsistente.

6.2. Trabajo a futuro

Las teorías de revisión de creencias han experimentado un fuerte desarrollo en años recientes, y se espera que dicho desarrollo continúe. Se espera que en el corto plazo aparezcan, entre otras cosas, diversos resultados teóricos concernientes al uso de lógicas no clásicas, por ejemplo, utilizar lógicas paraconsistentes (lógicas tolerantes a la inconsistencia) para modelar cambios en la base de conocimientos. Problemas de dinámica de credibilidad, modelos de difusión de opiniones, así como posibles aplicaciones en el campo de la psicología cognitiva, aplicaciones de modelos realistas para agentes económicos, en la investigación educativa, entre otros.

Algunas áreas de oportunidad en el desarrollo del trabajo presentado son:

- Modelar el operador de independencia en otro tipo de lógicas, como la lógica probablilistica para modelos de confianza, alguna lógica trivaluada, etc.

- Extender el algoritmo de revisión de creencias y conteo de modelos para mejorar la operación de contracción de la base de conocimiento.

- Implementar en un sistema web la operación de contracción de la base de conocimiento cuando ésta deja de ser consistente.

- Extender el modelado de problemas en el área de asistentes virtuales, por ejemplo, se puede modelar la difusión de opiniones a través de la interacción en una red social.

Apéndice A

Algoritmos implementados

En esta sección se describe la implementación de los algoritmos de revisión de creencias y de conteo de modelos de la base de conocimiento. Se realizaron aplicaciones web por la facilidad de tener acceso desde cualquier lugar a través de un navegador web.

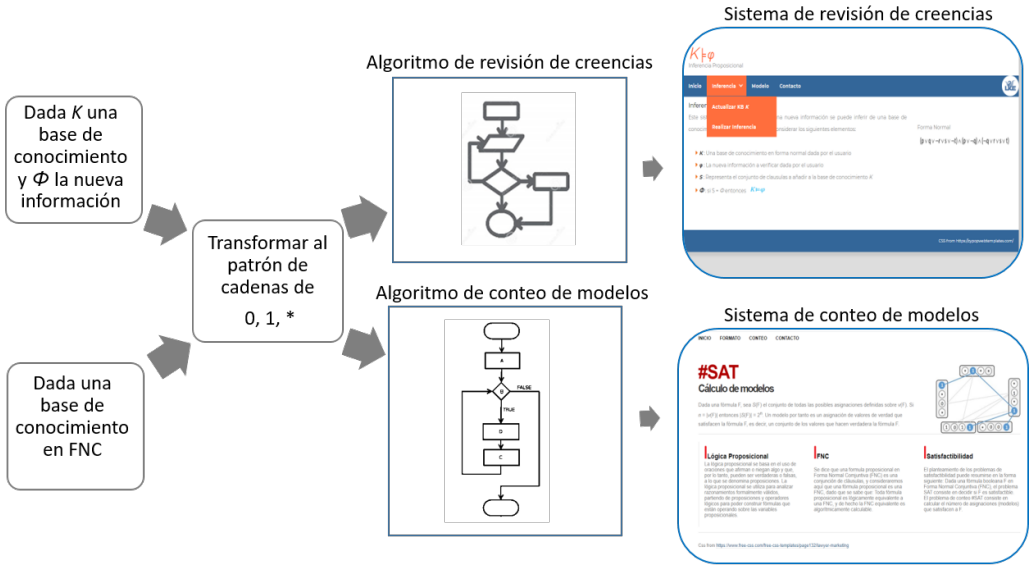


Figura A.1: Proceso de generación de los sistemas web

El procesamiento se realiza en el servidor donde se encuentra alojado el programa correspondiente y el resultado se presenta en el navegador. De manera temporal el sistema de revisión de creencias y de conteo de modelos se encuentran alojados en

la página personal del autor dentro del dominio de la Facultad de Ciencias de la Computación.
http://www.cs.buap.mx/∼pbello/sistemas.

En la Figura A.1 se describe el proceso general de funcionamiento de los sistemas implementados. En el caso del sistema de revisión de crencias se tiene como entrada la base de conocimiento (K) y la nueva información (ϕ), se realiza la transformación al patrón de ceros, unos y asteriscos, se aplica el algoritmo y obtenemos el resultado en el navegador web. Para el caso del sistema de conteo de modelos de la base de conocimiento, se tienen como entrada la base de conocimiento, posteriormente se trasforma usando el patrón descrito, se aplica el algoritmo conrrespondiente y se obtiene la salida en el navegador web.

A.1. Sistema de revisión de creencias

El sistema desarrollado de inferencia proposicional es una aplicación web basado en la lógica proposicional que implementa el algoritmo de revisión de creencias . El sistema permite realizar las siguientes funciones:

- Dar de alta una base de conocimiento K.

- Revisar inferencia desde un archivo de texto.

Actualizar KB K. Aquí se especifica la base de conocimiento necesaria para realizar el proceso de inferencia proposicional. La base de conocimiento debe estar alojada en un archivo de texto, donde la primera línea debe tener el número de cláusulas y el número de variables por cada cláusula. Además, en cada linea del archivo debe estar especificada cada clausula trasformada en el patrón de ceros, unos y *. Por ejemplo, sea el archivo baseCyS-2.txt con 4 cláusulas y 5 variables por cláusula:
4 5
10**0
110
101

110**

La Figura A.2 muestra la base de conocimiento cargada en el sistema de revisión de creencias.

Figura A.2: Base de conocimiento almacenada en el sistema

Realizar inferencia. para realizar inferencia se efectúa desde un archivo de texto. Después de cargar la base de conocimiento en el sistema, es necesario cargar la nueva información (ϕ) a revisar. La nueva información también esta almacenada en un archivo de texto con las mismas características que el archivo de la base de conocimiento. Consideremos el archivo FiCyS-2.txt con los siguientes datos:
4 5
1*1**
*10**
0011*
****0
La Figura A.3 muestra la nueva información en el sistema y el proceso de revisión de creencias. La salida del programa es un árbol de análisis donde a partir del segundo nivel representa el resultado de aplicar el algoritmo de revisión de creencias. La profundidad del árbol depende del número de cláusulas de la nueva información ϕ.

El resultado de la aplicación del algoritmo como se muestra en la Figura A.3 es

el conjunto S formado por cada una de las cláusulas de cada hoja del árbol. En este caso S=(11110), (01110), (01000), (00**0), (0011*), (0100*), (101*1), (1111*), que representa las clausulas necesarias para que se cumpla $K \models \phi$.

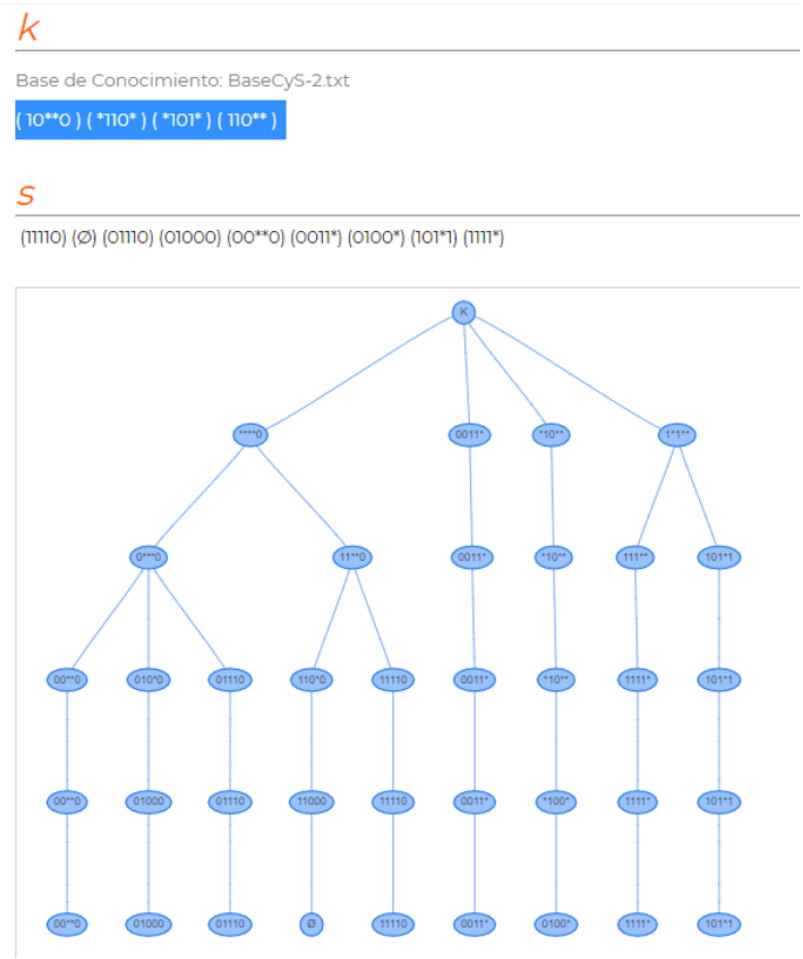


Figura A.3: Salida del sistema de revisión de creencias

A.2. Sistema de conteo de modelos

Como resultado del diseño del algoritmo de conteo de modelos se cuenta con un sistema en web implementado en el lenguaje de programación PHP. El sistema permite realizar pruebas del conteo de modelos de una base de conocimiento. Se

pueden realizar pruebas a través de archivos de texto, de forma manual o generando entradas aleatorias. El sistema de conteo de modelos se ejecuta en un servidor local.

En la Figura A.4 se muestra la pantalla principal del sistema web para el conteo de modelos de una base de conocimiento. La página inicial cuenta con una liga a la sección de "FORMATO" donde se especifica el formato que debe tener la base de conocimiento. En la liga de "CONTEO" se realizan las diferentes formas de realizar el conteo de modelos y en la liga de "CONTACTO" se describe los datos del autor del sistema.

Figura A.4: Pagina inicial del sistema de conteo de modelos

En la Figura A.5 se indica que podemos realizar el conteo de modelos de 3 maneras: datos por archivo, datos por teclado y datos aleatorios.

1. **Datos por archivo:** Se debe tener un archivo de texto, donde en la primera línea debe contener el número de cláusulas y el número de variables por cláusula, posteriormente se coloca una cláusula por línea usando la representación de 0,1 y asteriscos. Por ejemplo, considere el archivo de texto como archivo de entrada con 7 cláusulas y 9 variables:
   7 9

000******
*000*****
000**
000
****000**
*****000*
******000

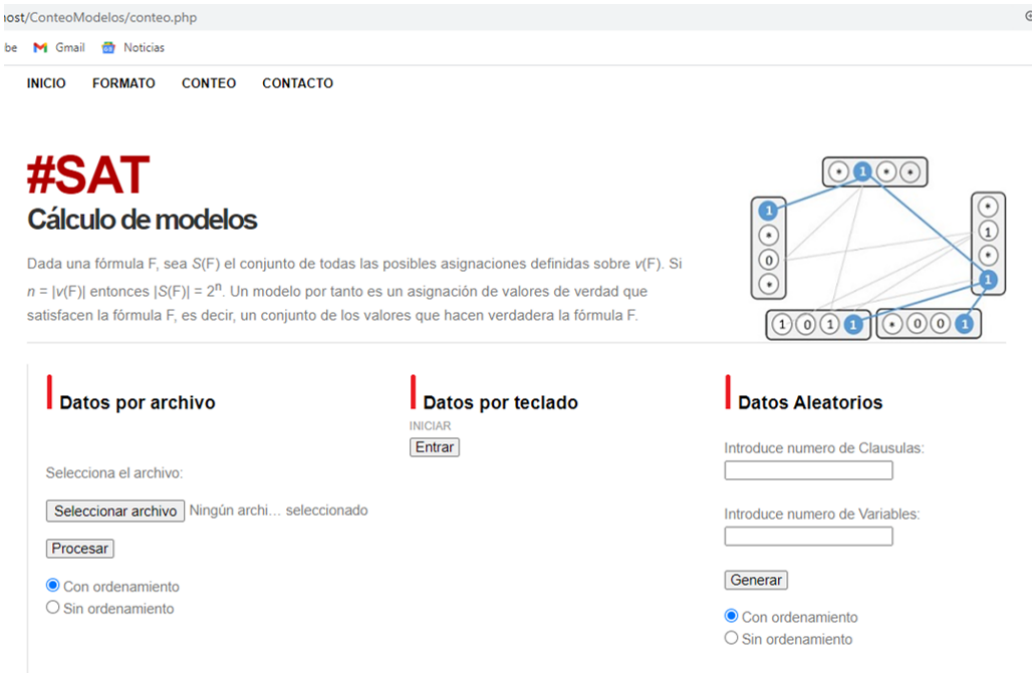


Figura A.5: Formas de realizar el conteo de modelos de la base de conocimiento

2. **Datos por teclado:** Aquí se solicita introducir el número de cláusulas e introducir el número de variables que tendrá cada cláusula. Posteriormente debe introducir una cláusula por renglón. La opción con ordenamiento o sin ordenamiento determina la forma de realizar el cálculo del conteo de modelos.

   En la Figura A.6 se muestra un ejemplo de ejecución con 5 cláusulas de 5 variables. En la parte izquierda de la figura se muestra el conteo de modelos con ordenamiento, y en la parte derecha se muestra el conteo de modelos sin

ordenamiento. Podemos notar que cuando aplicamos el ordenamiento se reduce el número de comparaciones y por tanto el cálculo de los modelos de la base de conocimiento es más rápido. En el caso del algoritmo de conteo de modelos con ordenamiento, se procesa primero la cláusula con más asteriscos, esto permite eliminar más modelos en la base de conocimiento y reduce el número de comparaciones a realizar.

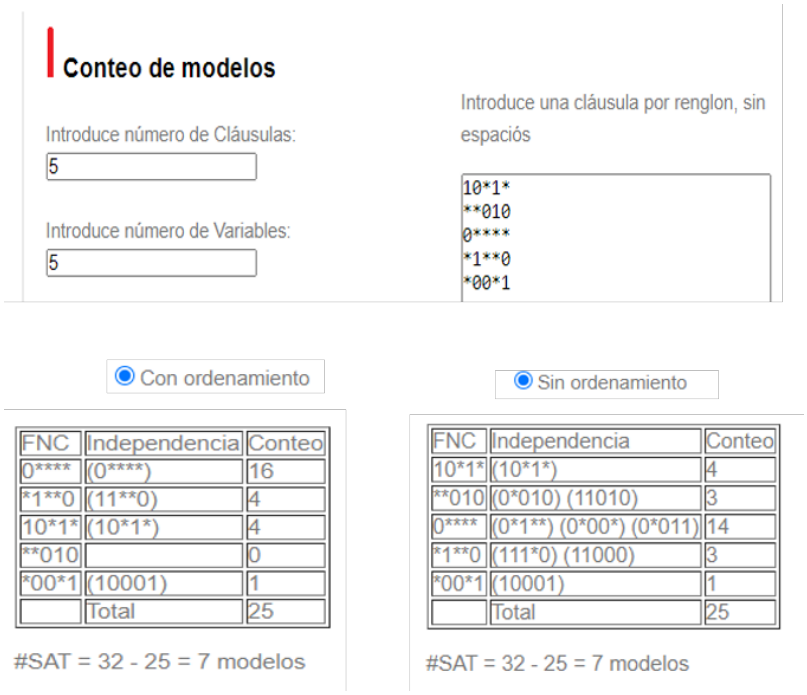
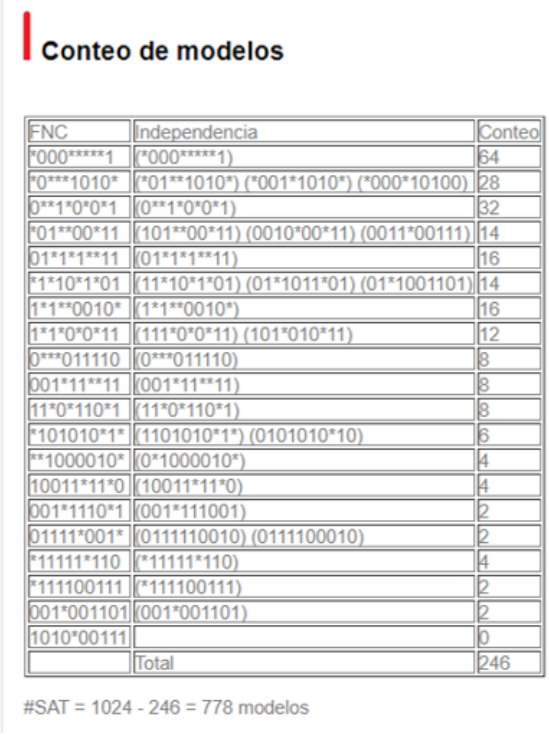


Figura A.6: Datos de teclado: con ordenamiento y sin ordenamiento

3. Datos aleatorios: El sistema de conteo de modelos permite realizar pruebas con instancias generadas de forma aleatoria. Para ejecutar pruebas aleatorias sólo es necesario introducir el número de cláusulas y el número de variables que tendrá cada cláusula. Como resultado se obtienen el conteo del número de modelos así como las cláusulas generadas en cada revisión. La Figura A.7 muestra un ejemplo de ejecución de forma aleatoria con 20 cláusulas y 10 variables. La primera columna muestra la base de conocimiento en FNC generada de forma aleatoria, la segunda columna indica la generación de cláusulas en el proceso de independencia. Y la tercera columna muestra el número de cláusulas.



Conteo de modelos

FNC	Independencia	Conteo
*000*****1	(*000*****1)	64
*0***1010*	(*01**1010*) (*001*1010*) (*000*10100)	28
0**1*0*0*1	(0**1*0*0*1)	32
*01**00*11	(101**00*11) (0010*00*11) (0011*00111)	14
01*1*1**11	(01*1*1**11)	16
*1*10*1*01	(11*10*1*01) (01*1011*01) (01*1001101)	14
1*1**0010*	(1*1**0010*)	16
1*1*0*0*11	(111*0*0*11) (101*010*11)	12
0***011110	(0***011110)	8
001*11**11	(001*11**11)	8
11*0*110*1	(11*0*110*1)	8
*101010*1*	(1101010*1*) (0101010*10)	6
**1000010*	(0*1000010*)	4
10011*11*0	(10011*11*0)	4
001*1110*1	(001*111001)	2
01111*001*	(0111110010) (0111100010)	2
*11111*110	(*11111*110)	4
*111100111	(*111100111)	2
001*001101	(001*001101)	2
1010*00111		0
	Total	246

#SAT = 1024 - 246 = 778 modelos

Figura A.7: Conteo de modelos de una base de conocimiento aleatoria

En la Tabla A.1 se muestran los resultados del cálculo de modelos de diversas entradas desde 10 hasta 300 cláusulas y diferentes números de variables por cláusula.

Tabla A.1: Diversas instancias generadas de forma aleatoria

Cáusulas	Variables	Modelos	Tiempo(Seg)
10	10	1024 - 128 = 896	0
20	10	1024 - 231 = 793	0
50	20	1048576 - 11741 = 1036835	0
100	20	1048576 - 20243 = 1028333	0
200	30	1073741824 - 1065208 = 1072676616	1
200	20	1048576 - 75492 = 973084	5
200	10	1024 - 992 = 32	1
300	40	1099511627776 - 41099096 = 1099470528680	1
300	30	1073741824 - 1808006 = 1071933818	3
300	20	1048576 - 65537 = 983039	22
300	15	32768 - 15565 = 17203	42
300	10	1024 - 1013 = 11	1

Bibliografía

[1] Belov A., Janota M., y Marques-Silva J. Algorithms for computing minimal equivalent subformulas. *Artificial Intelligence*, 266:309–326, 2014.

[2] Cook S. A. The complexity of theorem-proving procedures. *Proc. 3er Ann. ACM Symp. on theory of Computing, ACM*, 1(1):151–158, 1971.

[3] Darwiche A. On tractable counting of theory models and its application to truth maintenance and belief revision. *Applied Non-Classical Logics*, 11:11–34, 2001.

[4] Darwiche A. y Pearl J. On the logic of iterated belief revision. *Artificial Intelligence*, 89:1–29, 1997.

[5] Deagustini C. A., Teze J. C. L., Martinez V., Falappa M. A., y Simari G. R. Merging existential rules programs in multi-agent contexts through credibility accrual. *Information Sciences*, 555:236 – 259, 2021.

[6] Deagustini C. A., Martinez M. V., Falappa M. A., y Simari G. R. Belief base contraction by belief accrual. *Artificial Intelligence*, 275:78 – 103, 2019. ISSN 0004-3702. doi:https://doi.org/10.1016/j.artint.2019.05.002. URL http://www.sciencedirect.com/science/article/pii/S0004370219301249.

[7] Falappa M. A., Kern-Isberner G., Reis M. D. L., y Simari G. R. Prioritized and non-prioritized multiple change on belief bases. *Journal of Philosophical Logic*, 41:77–113, 2012.

[8] Falappa M. A., García A. L., Kern-Isberner G., y Simari G. R. On the evolving relation between belief revision and argumentation. *The Knowledge Engineering Review*, 26:35–43, 2011.

[9] Grove A. Two modellings for theory change. *Journal of Philosophical Logic*, 17:157–170, 1988.

[10] Haret A., Rummele S., y Woltran S. Merging in the horn fragment. *ACM Transactions on Computational Logic*, 18(1):6, 2017.

[11] Haret A. y Woltran S. Belief revision operators with varying attitudes towards initial beliefs. En *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, págs. 1726–1733. International Joint Conferences on Artificial Intelligence Organization, 2019. doi:10.24963/ijcai.2019/239. URL https://doi.org/10.24963/ijcai.2019/239.

[12] Herzig A., Lang J., y Marquis P. Propositional update operators based on formula/literal dependence. *ACM Transactions on Computational Logic*, 14:1–31, 2013.

[13] Nebel B. Belief revision and default reasoning: Syntax based approaches. *In Principles of knowledge representation and reasoning*, 1:417–428, 1991.

[14] Nebel B. How hard is it to revise a belief base. *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, 3:77–145, 1998.

[15] Selman B. *Tractable Default Reasoning*. Tesis Doctoral, Department of Computer Science University of Toronto, Toronto, Ont., Canada, Canada, 1990. UMI Order No. GAXNN-65936.

[16] Alchourróm C. y Makinson D. On the logic of theory change: contraction functions and their associated revision functions. *Theoria*, 48:14–37, 1982.

[17] Alchourrom C., Gardenfords P., y Makinson D. On the logic of theory change. partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50:510–530, 1985.

[18] Ansótegui C. y Manyá F. An introduction to satisfiability algorithms. *Artificial Intelligence*, 7(11):309–326, 2003.

[19] Beierle C., Eichhorn C., y Kern-Isberner G. On transformations and normal forms of conditional knowledge bases. *Springer International Publishing AG*, 1:488–494, 2017.

[20] Cresto C. Revisión de creencias y racionalidad. *Cuadernos del CIMBAGE*, 5:133–156, 2002.

[21] González R. C. Aproximación al concepto de inferencia desde dos modelos de comprensión: modelo estratégico y modelo de construcción e integración. *Literatura y Lingüística*, 35:297–314, 2017.

[22] Pons C., Rosenfeld R., y Smith C. *Lógica para informáticos*. Editorial de la Universidad de La Plata, Buenos Aires, Argentina, 1 ed$^{\underline{\text{ón}}}$., 2017.

[23] Jelenc D., Tamargo L., Gottifredi S., y García A. J. Credibility dynamics: A belief-revision-based trust model with pairwise comparisons. *Artificial Intelligence*, 293:103450, 2021.

[24] Jiang D., Li W., Luo J., Lou Y., y Liao Z. A representative model based algorithm for maximal contractions. *Science China Information Sciences*, 56:1–13, 2013.

[25] Lehmann D. Belief revision. En IJCAI, ed., *Proc. IJCAI'95*, págs. 1534–1540. 1995.

[26] Fermé E. Revisión de creencias. inteligencia artificial. *Revista Iberoamericana de Inteligencia Artificial*, 11(34):17–39, 2007.

[27] Fermé E. *On the Logic of Theory Change: Extending the AGM Model*. Tesis Doctoral, Royal Institute of Technology, Stockholm, Sweden, 2011.

[28] Fermé E., Mikalef J., y Taboada J. Credibility-limited functions for belief bases. *Journal of Logic and Computation*, 13(1):99–110, 2003.

[29] Fermé E. y Hansson S. O. Agm 25 years, twenty-five years of research in belief change. *Journal of Philosophical Logic*, 40:295–331, 2011.

[30] Boella G., Pigozzi G., y Torre L. V. AGM contraction and revision of rules. *Journal of Logic, Language and Information*, 25:273–297, 2016.

[31] De-Ita G. y Zacarias F. A model-based algorithm for propositional belief revisions. *IEEE Latin America Transactions*, 13:1055–1060, 2015.

[32] De-Ita G., Marcial R., Bello P., y Contreras M. Belief revisions between conjunctive normal forms. *Journal of Intelligent & Fuzzy Systems*, 34:3155–3164, 2018.

[33] De-Ita G., Marcial-Romero J. R., Hernández J. A., y Bello P. *Conocimiento y Razonamiento Computacional*. Academia Mexicana de Computación, México, 2018. ISBN 978-607-97357-3-9.

[34] Kern-Isberner G. y Huvermann D. What kind of independence do we need for multiple iterated belief change? *Journal of Applied Logic*, 22:91–119, 2016.

[35] López G., Jeder I., y Vega A. *Análisis y diseño de algoritmos: implementaciones en C y Pascal*. Alfaomega, Buenos Aires, Argentina, 1a ed. ed$^{\underline{\text{ón}}}$., 2009.

[36] Rúa M. G. y Sierra A. M. Algunas lógicas modales asociadas al razonamiento de agentes inteligentes. *Ingeniería y Ciencia*, 4(7):23–45, 1996.

[37] Cormen T. H., Leiserson C. E., Rivest R. L., y Stein C. *Introduction to Algorithms, Third Edition*. The MIT PRess, Cambridge, Massachusetts Londoc, England, 2009. ISBN 9780262033848.

[38] Drias H. A monte carlo algorithm for the satisfiability problem. *Lecture Notes in Computer Science*, 68:1415–1591, 2005.

[39] Katsuno H. y Mendelzon A. O. On the difference between updating a knowledge base and revising it. En *KR91 Cambridge*, págs. 387–394. Morgan Kaufmann, 1991.

[40] Papadimitriou C. H. *Computational Complexity*. Addison-Wesley Pub, University of California - San Diego, 1 ed$^{\underline{\text{ón}}}$., 1994.

[41] Velázquez H. Lógica deóntica: breve panorama de la cuestión. cuadrante phi. *Revista de estudiantes de filosofía*, 28:1–24, 2015.

[42] Zhang H. Sato: An efficient prepositional prover. *Automated Deduction—CADE-14, Springer Berlin Heidelberg*, 1:272–275, 1997.

[43] Levi I. *Gambling with Truth: an essay on induction and the aims of science.* Cambridge, MIT Press, 1977. ISBN 9780262620260.

[44] Delgrande J., Peppas P., y Woltran S. AGM-style belief revision of logic programs under answer set semantics. *Lecture Notes in Computer Science*, 8148:264–276, 2013.

[45] Delgrande J., Jin Y., y Pelletier F. J. Compositional belief update. *Journal of Artificial Intelligence Research*, 32:757–791, 2008.

[46] Stockmeyer L. J. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[47] Fagin R. K., Ullman J., y Vardi M. Y. Updating logical databases. *Advances in Computing Research*, 3:1–18, 1986.

[48] Lewis D. K. *Counterfactuals.* Harvard University Press, 1973. ISBN 9780674175402. URL https://books.google.com.mx/books?id=58uCQgAACAAJ.

[49] Satoh K. Nonmonotonic reasoning by minimal belief revision. *Institute for New Generation Computer Technology*, 358:147–170, 1990.

[50] Chu M. L. y Anbulagan. Heuristics based on unit propagation for satisfiability problems. *In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, 2(1):366–371, 1997.

[51] Harper W. L. Ramsey test conditionals and iterated belief change (a response to stalnaker). En *Series in Philosophy of Science*. The University of Western Ontario, 1976.

[52] Perrusssel L. y Marchi J. Prime forms and minimal change in propositional belief bases. *Ann. Math. Artif. Intelligence*, 59:1–45, 2010.

[53] Aiguier M., Atif J., Bloch I, y Hudelot C. Belief revision, minimal change and relaxation: A general framework based on satisfaction systems, and applications to description logics. *Artificial Intelligence*, 256:160–180, 2018.

[54] Cadoli M., Donini F. M., Liberatore P., y Schaerf M. Preprocessing of intractable problems. *Information and Computation*, 176:89–120, 2002.

[55] Dalal M. Investigations into theory of knowledge base revision. En AAAI, ed., *The Seventh National Conference on Artificial Intelligence*, tomo 1. 1988. Https://pdfs.semanticscholar.org/88a0/476a61b4cf2fb9c08a511d9ced44cb6f7564.pdf.

[56] Davis M., Logemann G., y Loveland D. A machine program for theorem-proving. commun. *ACM*, 5(7):394–397, 1962.

[57] Davis M. y Putnam H. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[58] Garapa M., Fermé E., y Reis M. D. L. Credibility-limited base revision: New classes and their characterizations. *Journal of Artificial Intelligence Research*, 69:1023–1075, 2020.

[59] Korpusik M., Lukaszewicz W., y Madalińska-Bugaj E. Consistency-based revision of structured belief bases. *Fundamenta Informaticae*, 136:381–404, 2015.

[60] Mouhoub M y Sadaoui S. Solving incremental satisfiability. *International Journal of Artificial Intelligence Tools*, 16:139–147, 2007.

[61] Moskewicz M.W., Madigan C.F., Zhao Y., Zhang L., y Malik S. Chaff: Engineering an efficient sat solver. En *In Proceedings of the 38th Annual Design Automation Conference, DAC '01*, págs. 530–535. ACM, 2001.

[62] Creignou N., Papini O., Pichler R., y Wolfran S. Belief revision whithim fragments of propositional logic. *Journal of Computer and System Sciences*, 80(2):427–449, 2014.

[63] Creignou N., Papini O., Rümmele S., y Woltran S. Belief merging within fragments of propositional logic. *ACM Transactions on Computational Logic*, 17(3):20, 2016.

[64] Creignou N., Ktari R., y Papini O. Belief update within propositional fragments. *Journal of Artificial Intelligence Research*, 61:807–834, 2018.

[65] Hooker J. N. Solving the incremental satisfiability problem. *The Journal of Logic Programming*, 15(1):177 – 186, 1993.

[66] Doubois O. Counting the number of solutions for instances of satisfiability. *Theoretical Computer Science*, 81:49–64, 1991.

[67] Hansson S. O. Belief contraction. *Journal of Symbolic Logic*, 59:845–859, 1988.

[68] Hansson S. O. Semi revision. *Journal of Applied Non-Classical Logics*, 7:151–175, 1997.

[69] Hansson S. O. A monoselective presentation of agm revision. *Studia Logica*, 103:1019–1033, 2014.

[70] Moguillanske M. O., Wassermann R., y Falappa M. A. Inconsistent-tolerant base revision through argument theory change. *Logic Journal of the IGPL*, 20:154–186, 2012.

[71] Bello P. Modelando las normas con revisión de creencias e inferencia. *Elementos*, 112:9–13, 2018.

[72] Bello P. y De-Ita G. Inference algorithm with falsifying patterns for belief revision. *Lecture Notes in Computer Science (LNCS) Springer*, 11524:347–356, 2019.

[73] Bello P. y De-Ita G. An algorithm to belief revision and to verify consistency of a knowledge base. *IEEE Latin America Transactions*, por aparecer:10, 2021.

[74] Bello P., De-Ita G., Contreras M., y Rodríguez M. Algoritmo para el conteo de modelos en FNC. *Research in Computing Science*, 149(11):147–158, 2020.

[75] Bello P., Rodríguez M., y De-Ita G. Extremal topologies for the merrifield-simmons index on dynamic trees. *Lecture Notes in Computer Science (LNCS) Springer*, por aparecer:10, 2021.

[76] Gardenfors P. Rules for rational changes of belief. En *En Philosophical essays dedicated to Lennart Aqvist on his fiftieth birthday, Pauli, T*, págs. 88–101. Departmento de Filosofia, Universidad de Uppsala, 1982.

[77] Liberatore P. The complexity of iterated belief revision. *Lecture Notes in Computer Science*, 1186:276–292, 1997.

[78] Liberatore P. Revision by history. *Journal of Artificial Intelligence Research*, 52:287–329, 2015.

[79] Liberatore P. y Schaerf M. The complexity of model checking for belief revision and update. En *Procc.Thirteenth Nat. Conf. on Artificial Intelligence AAAI*. AAAI, 1996.

[80] Liberatore P. y Schaerf M. Belief revision and update: Complexity of model checking. *Journal of Computer and System Sciences*, 62:43–72, 2001.

[81] Marques-Silva J. P. y Sakallah K. A. Grasp: a search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[82] Peppasa P., Willians M. A., Choprac S., y N. Foo. Relevance in belief revision. *Artificial Intelligence*, 229:126–138, 2015.

[83] Pozos-Parra P., Chávez-Bosquez O., y McAreavey K. A belief merging tool for consensus support. *Journal of Intelligent & Fuzzy Systems*, 34:3199–3210, 2018.

[84] Booth R. y Hunter A. Trust as a precursor to belief revision. *Journal of Artificial Intelligence Research*, 61:699–722, 2018.

[85] Booth R. y Chandler J. From iterated revision to iterated contraction: Extending the harper identity. *Artificial Intelligence*, 277:103171, 2019.

[86] Booth R. y Chandler J. On strengthening the logic of iterated belief revision: Proper ordinal interval operators. *Artificial Intelligence*, 285:103289, 2020.

[87] Booth R., Meyer T., y Varzinczak I. J. Next steps in propositional horn contraction. En IJCAI, ed., *Proc. 21st. Int. Join the conference on artificial Intelligence-IJCAI*, tomo 21. 2009. Https://researchspace.csir.co.za/dspace/handle/10204/3597.

[88] Caridroit R., Konieczny S., y Marquis P. Contraction in propositional logic. *International Journal of Approximate Reasoning*, 80:428–442, 2017.

[89] Khardon R. y Roth D. Reasoning with models. *Artificial Intelligence*, 87:187–213, 1996.

[90] Rosenfield R. y Irazábal J. *Computabilidad, complejidad computacional y verificación de programas*. Universidad Nacional de la Plata, Argentina, 1a ed. ed$^{\underline{\text{ón}}}$., 2013.

[91] Zuleta H. R. Lógica deóntica y verdad. *Análisis filosófico XXVI*, 1:115–133, 2006.

[92] Luan S., Dai G., y Magnani L. An approximate approach to belief revision. *Logic Journal of the IGPL*, 20:486–496, 2010.

[93] Mcculloch W. S. y Pitts W. Grasp: a search algorithm for propositional satisfiability. *Bulletin of Mathematical Biology*, 52:99–115, 1990.

[94] Aravanis T., Peppas P., y Williams M. A. Full characterization of parikh's relevance-sensitive axiom for belief revision. *Journal of Artificial Intelligence Research*, 66:765–792, 2019.

[95] Aravanis T., Peppas P., y Williams M. A. An investigation of parametrized difference revision operators. *Annals of Mathematics and Artificial Intelligence*, 89:7–28, 2021.

[96] Eiter T. y Gottlob G. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[97] Liu W., Pozos P., y Perrussel L. Dalalś revision without hamming distance. En MICAI, ed., *In 12th Mexican International Conference on Artificial Intelligence*, págs. 41–53. 2013.

[98] Zhuan Z.Q. y Pagnucco M. Horn contraction via epistemic entrenchment. logics in artificial intelligence. *Lecture Notes in Computer Science*, 6341:339–351, 2010.