# Diseño de Algoritmos Combinatorios para # SAT y su Aplicación al Razonamiento Proposicional

Carlos Guillén, Dr. Aurelio López, Dr. Guillermo De Ita

Reporte Técnico No. CCC-05-005
11 de noviembre de 2005



Luis Enrique Erro 1
Sta. Ma. Tonantzintla,
72840, Puebla, México.

# Diseño de Algoritmos Combinatorios para $\#SAT$ y su Aplicación al Razonamiento Proposicional

MC. Carlos Guillén G. Dr. Aurelio López L. Dr. Guillermo De Ita L.

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
E-mail: {cguillen,allopez,gdeita}@inaoep.mx

## 1 Resumen

Uno de los problemas más famosos en las ciencias de la computación, específicamente en el campo de la complejidad computacional es el problema de la satisfactibilidad de restricciones. Este problema fue introducido por Cook [18], y desde entonces se ha analizado en diversas áreas y se ha relacionado con una amplia cantidad de aplicaciones computacionales.
Dentro de los problemas de satisfactibilidad de restricciones, se encuentra el problema de decisión, que consiste en determinar si una fórmula es satisfactible o no. El problema de conteo asociado, es saber el número de asignaciones que satisfacen a una fórmula Booleana, conocido como el problema $\#SAT$.
Existen muchos problemas no completamente resueltos que se derivan del problema $\#SAT$, por lo que se tiene un constante interés por parte de los investigadores de diversas áreas, por encontrar diferentes maneras de resolver el problema $\#SAT$, en parte, debido al impacto que tendría una buena solución de éste en muchas otras disciplinas y aplicaciones.
La investigación a realizar recae sobre el diseño y desarrollo de algoritmos para $\#SAT$, ésta involucra como preguntas claves, saber si existe un algoritmo que lo resuelve eficientemente, cuál es la frontera entre lo computable en tiempo polinomial y lo computable en tiempo exponencial y qué impacto tiene la solución de problemas de satisfactibilidad de restricciones en el área del razonamiento proposicional.
En este reporte presentamos el estado del arte de los resultados más relevantes en esta dirección así como algunos avances dirigidos al diseño y desarrollo de algoritmos eficientes del problema $\#SAT$, por lo que consideraremos los problemas abiertos que se encuentran actualmente en el borde entre lo que es reconocido como la clase de complejidad FP (funciones que pueden calcularse computacionalmente en tiempo polinomial) y la clase de complejidad $\#P$. Con el objetivo de seguir aclarando y fortaleciendo las diferencias entre lo computable eficientemente y la clase $\#P$ (funciones computables de forma determinista en tiempo de orden exponencial).

## 2 Introducción

En el área de las ciencias computacionales surgen problemas de investigación donde se sabe que la complejidad en tiempo de los procedimientos propuestos para resolverlos es de naturaleza exponencial y hasta la fecha, no se han construido métodos eficientes para resolver estos problemas. Cuando algunos de es-

tos problemas son formalizados encontramos una fuerte vinculación con el problema de Satisfactibilidad (SAT), de Máxima Satisfactibilidad (MAXSAT) y el problema de conteo de Satisfactibilidad (#SAT). En términos simples, el problema SAT consiste en saber si una fórmula Booleana dada, es verdadera para alguna asignación de sus variables, mientras que en el problema #SAT nos interesa contar el número de asignaciones sobre las variables que hacen verdadera a la fórmula Booleana.
Los problemas SAT y #SAT son problemas centrales de la lógica matemática y de la teoría de la computación. Estos son fundamentales para la solución de varios problemas en razonamiento automático, diseño y manufactura asistida por computadora, planificación, visión computacional, bases de datos, robótica, diseño de circuitos integrados, arquitectura y redes de computadoras, entre otros [41, 81, 57, 63].
Será de particular interés para nosotros el problema #SAT, porque además de su vinculación con muchos otros problemas, la solución de éste lleva como consecuencia la solución de su correspondiente problema de decisión SAT. Sabemos que en su completa generalidad, #SAT es un problema de la clase #P-completo, incluso versiones más débiles del problema siguen perteneciendo a la misma clase.
Resulta natural la identificación de clases de fórmulas Booleanas donde el problema #SAT pueda ser resuelto en tiempo polinomial. Como veremos más adelante, se conoce muy poco sobre las restricciones a #SAT donde su solución sea eficiente, incluso si optamos por métodos aproximados de conteo.
Otra línea de investigación relacionada con nuestro proyecto está dirigida a la identificación de estructuras algebraicas contenidas en las relaciones definidas por fórmulas Booleanas, y que permitan un cómputo polinomial al tratar el problema #SAT. En esta dirección, se han desarrollado los Teoremas de Dicotomía [71, 19, 13, 50], que en términos generales tratan de establecer la frontera entre lo tratable y lo intratable para SAT y #SAT.
Dentro de las múltiples aplicaciones de #SAT, se encuentran las aplicaciones sobre el razonamiento proposicional, el cual constituye una pieza fundamental de la Inteligencia Artificial.
También existen aplicaciones donde se requieren sistemas que puedan decidir por sí mismos lo que necesitan hacer para satisfacer sus objetivos propuestos. Tales sistemas de cómputo son conocidos como $Agentes$. Los agentes que operan de forma robusta ante cambios repentinos, impredecibles, o en medios abiertos donde existe una posibilidad de que las acciones puedan fallar, son conocidos como $Agentes\ Inteligentes$. Aunque no existe una definición universalmente aceptada, para muchos propósitos un agente inteligente se puede considerar como un sistema de cómputo que es capaz de una acción autónoma, $flexible$ cuando se encuentra con objetivos planeados, donde flexible significa tres cosas: reactividad (percepción y respuesta a su medio ambiente), proactividad (comportamiento dirigido para formar iniciativas), y habilidad social (interacción con otros agentes) [82].
El paradigma de agentes inteligentes ha replanteado a la comunidad de investigadores, el diseño y desarrollo de nuevos algoritmos que permitan realizar razonamiento automático [53, 30, 31]. Es claro que el avance de los algoritmos de razonamiento automático se basan en el diseño de algoritmos eficientes para resolver los problemas de Satisfactibilidad (SAT) y sus versiones de optimización Máxima Satisfactibilidad (MAXSAT) y conteo ($\#SAT$). El problema MAXSAT consiste en encontrar una asignación que satisface el número máximo de cláusulas de una fórmula Booleana dada. Aún cuando es reconocida la complejidad exponencial de estos problemas, es un problema abierto el reconocer las subclases de fórmulas donde tales problemas pueden resolverse eficientemente. El identificar estas subclases influye directamente en el desarrollo de algoritmos que permitan simular el razonamiento de un agente inteligente de forma eficiente, que permitan revisar-actualizar su base de creencias y calcular eficientemente el grado de creencia que el agente tendrá sobre nueva información.
Este documento se deriva de una propuesta para un proyecto de investigación orientada al diseño y desarrollo de algoritmos eficientes para las diferentes restricciones del problema #SAT que se encuentran actualmente

en el borde entre lo que es reconocido como la clase de complejidad FP (funciones que pueden calcularse computacionalmente en tiempo polinomial) y la clase de complejidad #P. Otra línea de investigación se orienta a la identificación de estructuras algebraicas (sobre relaciones y fórmulas Booleanas) que nos ayuden a seguir aclarando y fortaleciendo las diferencias entre lo computable eficientemente y la clase $\#P$.

## 3 Estado del Arte

Los problemas de satisfactibilidad son estudiados intensivamente en las ciencias computacionales, debido a que son sin duda, los problemas más importantes en el campo de la complejidad computacional [67], y una de las aplicaciones interesantes es cuando contribuyen significativamente a simplificar un problema que por naturaleza incluye procesos exponenciales. Existen diversas líneas de investigación en torno a estos problemas y las aplicaciones son muy variadas.
Nuestro objetivo principal aquí es mostrar la línea de investigación a seguir y el estado actual de los resultados más relevantes, así como centrarnos en una sola línea de aplicaciones. Para esto, hemos dividido esta sección en los siguientes puntos que consideramos delimitan nuestro proyecto de investigación.

### 3.1 El Problema $\#SAT$

El problema de satisfactibilidad (SAT) fue introducido en el campo de la complejidad computacional en 1971 por Cook [18] (actualmente este problema aparece extensivamente en un rango diverso de áreas y en una variedad de formas [70, 68, 17, 76, 1]. Una forma simple de enunciar el problema SAT es: dada una fórmula Booleana, ¿existe una asignación que la satisface?. Se sabe que este problema pertenece a la clase NP-completo (problemas de decisión en NP tal que todo otro problema en NP se puede transformar en tiempo polinomial en uno de éstos).
Nosotros estamos interesados en el correspondiente problema de conteo: #SAT, esto es, dada una fórmula Booleana, determinar el número de asignaciones que la satisfacen. Podemos plantear el problema #SAT de la siguiente manera:

**Problema: #SAT**

**Input:** $\Sigma$ una fórmula Booleana.

**Output:** $\mid SAT(\Sigma) \mid$.

Donde $\mid SAT(\Sigma) \mid$ es la cardinalidad del conjunto de asignaciones que satisfacen a $\Sigma$.
En su completa generalidad, #SAT es un problema difícil, de hecho este problema se encuentra ubicado en la clase $\#P$-completo (problemas de conteo que pertenecen a $\#P$ tales que cualquier otro problema en $\#P$ se puede reducir en tiempo polinomial a uno de éstos), aún más se ha demostrado que éste problema es difícil de aproximar [75]. Debido a esto, los intentos de resolverlo se hacen sobre instancias específicas de fórmulas que nos restringen el alcance del problema. Existen diferentes formas de restringir un problema y la elección de una de ellas resulta ser arbitraria.
Un primer caso de simplificación en las fórmulas de entrada al problema, es considerar $Formas\,Conjuntivas$ (una forma conjuntiva es una conjunción de cláusulas y una cláusula es una disyunción de literales). En lo que sigue, $\Sigma$ denota una fórmula Booleana en forma conjuntiva.

**Restricciones HORN y MON**. Una cláusula es $HORN$ si contiene a lo más una variable no negada, y

es monótona ($MON$) si toda literal es no negada.
**Restricción sobre el número de literales**. Si cada cláusula de $\Sigma$ tiene a lo más k literales, el problema de contar el número de asignaciones que satisfacen $\Sigma$ se considera un problema en $\#kSAT$ ($\#kHORN$ o $\#kMON$ si es $HORN$ o $MON$, respectivamente).
**Restricción sobre el número de ocurrencias por variable**. Si cada variable en la fórmula $\Sigma$ aparece a lo más k veces, el problema de contar el número de asignaciones que satisfacen $\Sigma$ se considera un problema en $\#k\mu - SAT$.

Una clasificación de los resultados obtenidos en el desarrollo de algoritmos para #SAT, puede describirse de acuerdo a si estos algoritmos son exactos o aproximados (un algoritmo es exacto si para toda instancia de entrada siempre encuentra la solución buscada, un algoritmo es aproximado si la solución que se halla está dentro de un intervalo cercano al valor o a los valores que se buscan). Se tienen a la fecha cuatro clases de resultados:

1) **Resultados obtenidos al considerar algoritmos exactos en tiempo polinomial**. Existen pocos problemas en la jerarquía de resticciones que hemos elegido para los cuales se conoce que el conteo exacto del número de modelos para una fórmula Booleana se puede realizar en tiempo polinomial, de hecho, se tienen sólo dos resultados relevantes. El primero se debe a Vadhan [77], quien demostró que $2\mu - 2MON$ puede ser resuelto usando fórmulas de recurrencia. El segundo se debe a Roth [69], quien generaliza este resultado a $2\mu - 2SAT$. Sus algoritmos utilizan técnicas combinatorias que se han mejorado considerando el problema como un problema de orientación de grafos libres de sumideros [70]. Recientemente, nosotros hemos mostrado la existencia de una clase polinomial para #2SAT [27, 46], esta clase contiene a las halladas por Vadhan y Roth (caso (2,2$\mu$-FC)) y la extiende considerando la topología estructural del grafo $G_\Sigma$ definido por la fórmula $\Sigma$. Hemos probado que si $G_\Sigma$ es acíclico o contiene ciclos disjuntos entonces $\#SAT(\Sigma)$ puede computarse en tiempo polinomial. Esta nueva clase polinomial no pone restricciones sobre el número de ocurrencias de una variable a la fórmula, lo cual nos lleva a una nueva línea de investigación para resolver #SAT.

2) **Resultados obtenidos al considerar algoritmos exactos en tiempo exponencial** . Hay muchos resultados obtenidos para #SAT que están en la clase de complejidad #P-completo. Comenzando por los resultados de Valiant [80], donde se demuestra que dada una fórmula Booleana $\Sigma$, el contar el número de asignaciones que la satisfacen es un problema $\#P$-completo y el problema clásico de seguridad de redes es $\#P$-difícil.
Roth [69] demostró usando una simple técnica de reescritura, que uno puede a través de una reducción parsimoniosa de $\#2MON$ a $4\mu - 2HORN$, llegar a que este último problema es $\#P - completo$. (Si un problema de conteo $A$ se puede reducir a un problema de conteo $B$ en tiempo polinomial, y existe una biyección entre las soluciones de $A$ y $B$, entonces decimos que hay una reducción parsimoniosa entre $\#A$ y $\#B$).
Vadhan [77] demostró que $\#4\mu - 2MON$ es $\#P - completo$; más tarde, fortaleció este mismo resultado en el escenario de conjuntos independientes, para demostrar que también el conteo de conjuntos independientes en grafos bipartitos planos de grado máximo 4 es $\#P - completo$ [78]. (Un conjunto independiente de un grafo es un subconjunto de vértices del grafo que no contienen ambos puntos extremos de cualquier arista, un grafo bipartito es un grafo en el cual los vértices pueden ser particionados en dos subconjuntos, tal que ninguna arista tiene puntos extremos en el mismo subconjunto y un grafo es plano si existe un mapeo inclusivo del grafo al plano donde ningún par de aristas se intersecta).

Dyer y Greenhill [29] mejoran los resultados de Vadhan mostrando que $\#3\mu - 2MON$ es $\#P - completo$. Finalmente, Bubley and Dyer [11] demuestran que $\#2\mu - SAT$ es $\#P - completo$ y establecen que $\#2\mu - MON$ también es $\#P - completo$.

3) **Resultados para $\#$SAT considerando algoritmos aproximados y que corren en tiempo polinomial**. Existen pocos resultados en esta área y la mayoría de estos se apoyan en cadenas de Markov, métodos Monte Carlo, y en particular, se apoyan en técnicas de acoplamiento para pruebas de mezclado rápido. Entre los resultados más destacados podemos citar los trabajos de: Dyer y Russ [29], Luby and Vigoda [62], Dyer and Greenhill [33]. En estos trabajos se demuestra la existencia de un esquema de aproximación aleatoria completamente polinomial (fpras) para los problemas: $\#2\mu - SAT$, $\#4\mu - 2MON$, $\#3\mu - 3MON$. (Un fpras es un algoritmo aleatorio que corre en tiempo acotado por un polinomio en el número de las entradas y en el inverso de un parámetro de error con el que se acota la precisión de la solución encontrada).

4) **Resultados para $\#$SAT considerando algoritmos aproximados y que corren en tiempos exponenciales**. Aquí también existen muy pocos resultados para $\#$SAT. Entre estos resultados figuran los trabajos de Sinclair [75], sobre las implicaciones que tendría la existencia de un fpras para $\#2MON$, y los trabajos de Russ [70], donde demuestra que $\#k\mu - 2MON$ es NP-difícil, para $k \geq 3$.

### 3.1.1 Problemas abiertos que se encuentran en el borde de las clases FP, $\#$P y $\#$P-completo

Dentro de las versiones de $\#SAT$ que aún permanecen sin determinarse la clase de complejidad en la que se encuentran, están $\#2\mu - 3MON$ y $\#2\mu - kSAT$ con $k > 2$ (ver tablas 1 y 3). En el caso de soluciones aproximadas, no se sabe si exite un $fpras$ para $\#5\mu - 2MON$, $\#4\mu - 3MON$, o $\#3\mu - 4MON$, y si $\#24\mu MON$ es difícil de aproximar.

En la siguientes tablas, expresadas previamente por Russ [70], se muestra una clasificación sobre el problema $\#SAT$, incluyendo los casos en que permanece abierta la determinación de su clase de complejidad (marcada con una interrogación ?). La parte izquierda de cada rectángulo corresponde al conteo exacto, la parte derecha al conteo aproximado, el número 1 significa fácil, el 0 significa difícil y el signo de interrogación significa que aún no se sabe.

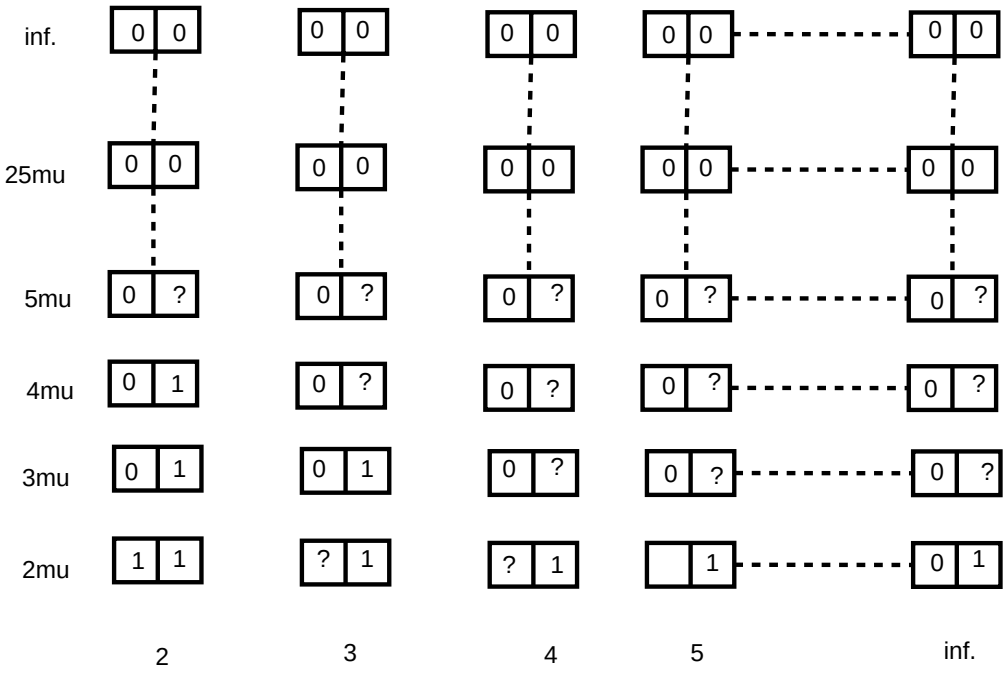


Tabla 1. Jerarquía de restricciones $\#MON$

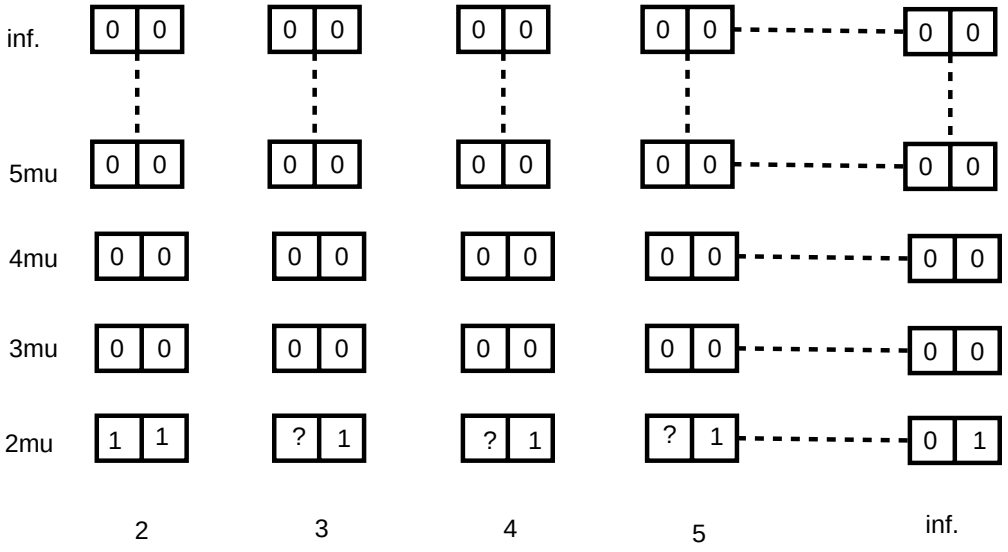


Tabla 2. Jerarquía de restricciones $\#HORN$

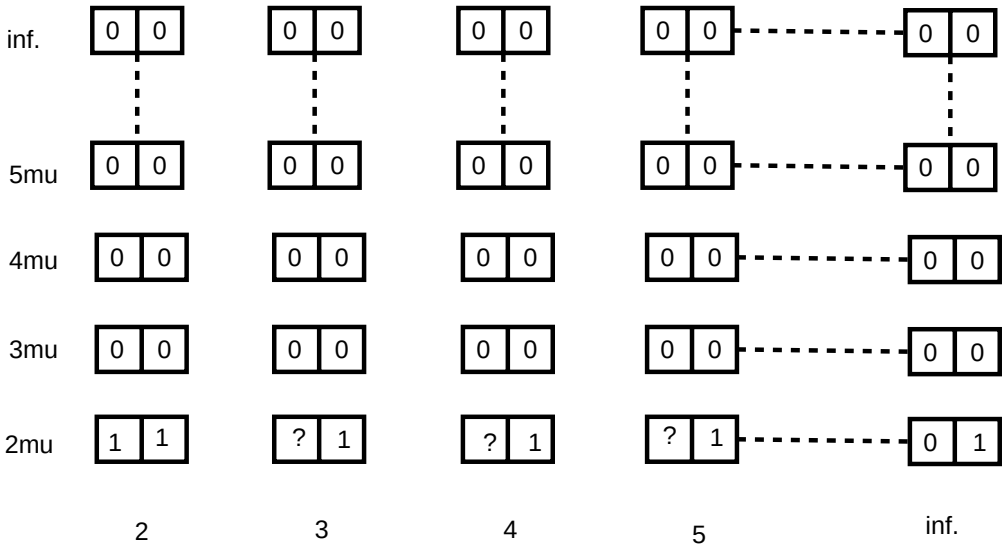


Tabla 3. Jerarquía de restricciones $\#SAT$

## 4 El Problema MAXSAT

El problema de máxima satisfactibilidad (MAXSAT), consiste en determinar una asignación que satisfaga el número máximo de cláusulas de una fórmula Booleana dada. Este problema ha retomado considerable interés debido a que constituye un problema fundamental para muchos problemas prácticos en ciencias de la computación e ingeniería [49, 83]. El problema MAXSAT es NP-completo, aún si las cláusulas tienen a lo más dos literales (MAX-2-SAT). Muchos de los métodos para MAXSAT estan basados en algoritmos de aproximación [23] y algunos basados en métodos de ramificación y corte [49, 10, 6, 51, 52, 44, 66] y otros en la transformación de MAXSAT en SAT [83]. Con respecto a los algoritmos exactos, un procedimiento típico de ramificación y corte ha sido la aplicación del método de Davis y Putman-Logenmann-Loveland (DyPLL)[25].

**Método DyPLL.**

• Si $\Sigma$ es el conjunto vacío, se retorna ”satisfactible”.
• Si $\Sigma$ contiene una cláusula vacía, se regresa ”no satisfactible”.
• Si $\Sigma$ tiene una cláusula unitaria, asignarle a su variable el valor de verdad que la satisface, y regresar el resultado de llamar el algoritmo Davis-Putman en la fórmula simplificada.
• Se selecciona del conjunto de variables una a la que no se le haya asignado valor, se le asigna un valor de verdad y se llama al procedimiento sobre la fórmula que haya resultado. Si el proceso resulta no satisfactible [1], se le asigna el valor opuesto a la variable seleccionada.

Cada vez que se selecciona una literal se divide el problema en dos, ya que son los posibles valores que

[1]Notese que en este caso se puede llevar un contador que nos indique el número máximo de cláusulas que se han satisfecho a ese nivel, adición que se utiliza para aplicar DyPLL a MAXSAT.

se pueden tomar, por lo que, esto se puede representar mediante un árbol binario de búsqueda, donde cada nodo representa una literal $l_n$, que ramifica con los dos posibles valores de verdad para la siguiente literal a escoger.

## 5 Teoremas de Dicotomía

En lo referente a los teoremas de dicotomía se tienen dos resultados relevantes: el de Schaefer [71], y el de Creignou [19].
Schaefer prueba un teorema de dicotomía para problemas de satisfactibilidad generalizada. En su famoso artículo: "The Complexity of Satisfiability Problems" [71], Schaefer define un número infinito de problemas de satisfactibilidad proposicional (en la terminología actual éstos problemas son llamados problemas de satisfacción de restricciones Booleanas [50]), y demuestra que todos estos problemas o bien están en $P$ o bien son NP-completos, al mismo tiempo, propone un criterio simple para determinar cuál de estos casos cumplen las fórmulas a considerar.
El Teorema de Dicotomía Scheafer para SAT, ha detonado actualmente en un conjunto de trabajos orientados a establecer un teorema análogo para los problemas de conteo de satisfacción de restricciones y problemas relacionados [8, 50, 9, 13, 20, 21]. Entre estos trabajos destaca el resultado de Creignou [19, 20], quien demuestra un teorema de Dicotomía para problemas de conteo de satisfactibilidad generalizada. El teorema de Dicotomía de Creignou, establece que si todas las relaciones lógicas involucradas sobre un problema de conteo de satisfactibilidad generalizada son afines entonces el número de asignaciones de satisfactibilidad de este problema puede ser computado en tiempo polinomial, de otra manera el problema es $\#P - completo$.
Debido a la importancia de los trabajos de Schaefer y Creignou a continuación enunciamos sus resultados principales. Antes, introducimos algunos conceptos utilizando la terminología y notación de [50]. Una **restricción** $C$ es una función Booleana $C : \mathbb{B}^k \longrightarrow \mathbb{B}$, donde $\mathbb{B} = \{0, 1\}$ y $k > 0$ es la aridad de C. Si C es una restricción de aridad $k$ y $z_1, z_2, \cdots, z_k$ son variables, entonces $C(z_1, z_2, \cdots, z_k)$ es la **aplicación de la imagen de C**, y si para $i \in \{1, \cdots, k\}$, $z_i$ es una variable o constante, entonces $C(z_1, z_2, \cdots, z_k)$ es la **aplicación de la restricción de C con constantes**.
Los problemas de satisfactibilidad generalizada de Schaefer se pueden definir formalmente usando la terminología de restricciones, de la siguiente manera:

i) **SAT(**$S$**)** es el problema de decidir si un conjunto dado $S = \{C_1, ..., C_m\}$ de aplicaciones de restricción es satisfactible, es decir si existe una asignación a las variables de $S$ que satisfacen a toda restricción en $S$.
ii) **SAT**$_c$**(**$S$**)** es el problema de decidir si un conjunto dado $S$ de aplicaciones de restricción con constantes en $S$ es satisfactible.

Una restricción C es:

1) **0-válida** si $C(0, 0, \cdots, 0) = 1$,
2) **1-válida** si $C(1, 1, \cdots, 1) = 1$,
3) **Horn o débilmente negativa** si $C$ es equivalente a una fórmula en FNC donde cada cláusula tiene a lo más una variable positiva, por ejemplo, $C(x, y, z) = (x \rightarrow y) \wedge (z \rightarrow (y \wedge \sim x))$ es equivalente a $(\sim x \vee y) \wedge (\sim x \vee y \vee \sim z)$,
4) **anti-Horn o débilmente positiva** si $C$ es equivalente a una fórmula en FNC donde cada cláusula tiene a lo más una variable negativa, por ejemplo, $C(x, y, z) = (x \rightarrow y) \wedge (z \rightarrow (x \vee y))$ es equivalente a $(\sim x \vee y) \wedge (x \vee y \vee \sim z)$,

5) **afín** si $C$ es equivalente a algún sistema de ecuaciones lineales sobre el campo GL(2), esto es, $C$ es equivalente a la conjunción de fórmulas de la forma:

$$\xi_1 \oplus \xi_2 \oplus \cdots \oplus \xi_n = \begin{cases} 1 \\ 0 \end{cases}$$

donde las $\xi_i$ son literales para cada $i = 1, ..., n$, y $\oplus$ denota x-or, esto es, $x \oplus y = (x \wedge \sim y) \vee (\sim x \wedge y)$, por ejemplo, $C(x, y) = (x \vee y) \wedge (\sim x \vee \sim y)$ es equivalente a $x \oplus y = 1$,
6) **biyuntiva** si $C$ es equivalente a una FNC que tiene a lo más dos literales en cada conjuntando ($2FNC$), por ejemplo, $C(x, y) = (x \equiv y) \wedge (y \equiv z)$ es equivalente a $(x \vee \sim y) \wedge (y \vee \sim x) \wedge (y \vee \sim z) \wedge (z \vee \sim y)$,
7) **complementativa** si para todo $s \in \mathbb{B}^k$, $C(s) = C(\bar{s})$, donde $k$ es la aridad de $C$ y $\bar{s} = (1 - s_1)(1 - s_2) \cdots (1 - s_k)$ para $s = s_1 s_2 \cdots s_k$, por ejemplo, $C(x, y) = (x \wedge y) \vee (\sim x \wedge \sim y)$ cumple que $C(x, y) = C(\bar{x}, \bar{y})$.

**Observación**. Una forma simple de verificar que una restricción es afín es a través del siguiente resultado derivado del álgebra lineal.

**Proposición 1** *Una restricción $C : \mathbb{B}^k \longrightarrow \mathbb{B}$ es afín si y sólo si, para toda terna $s_1, s_2, s_3 \in \mathbb{B}^k$ tal que $C(s_i) = 1$, $i = 1, 2, 3$, se cumple $C(s_1 \oplus s_2 \oplus s_3) = 1$.*

Sea $S$ un conjunto finito de restricciones, decimos que $S$ es **0-válido**, **1-válido**, **Horn**, **anti-Horn**, **biyuntivo**, **afín**, o **complementativo** si toda restricción $C \in S$ es 0-válida, 1-válida, Horn, anti-Horn, biyuntiva, afín, o complementativa, respectivamente.

El siguiente teorema es el resultado principal de Schaefer, el cual caracteriza la complejidad de $SAT(S)$ y $SAT_c(S)$.

**Teorema 1** ***(Dicotomía de Schaefer para SAT)***.
*1. Si $S$ es 0-válido, 1-válido, Horn, anti-Horn, biyuntivo o afín, entonces SAT(S) es decidible en tiempo polinomial. De otra manera SAT(S) es NP-completo.*
*2. Si $S$ es Horn, anti-Horn, biyuntivo o afín, entonces $SAT_c(S)$ es decidible en tiempo polinomial. De otra manera $SAT_c(S)$ es NP-completo.*

El problema de conteo de satisfactibilidad generalizada de Creignou se puede definir formalmente usando la terminología restricción de la siguiente manera:

$\#$**SAT(**$S$**)** es el problema de contar el número de asignaciones que satisfacen un conjunto dado $S$ de restricciones, es decir cuántas asignaciones a las variables de $S$ satisfacen toda restricción en $S$.

A continuación presetamos un segundo resultado que caracteriza la complejidad de las aplicaciones restricción cuantificadas. Antes introducimos algunos conceptos.
El problema de decidir si una fórmula Booleana cuantificada es verdadera se denota por **QBF**. Se sabe que este problema es PSPACE-completo aún si éste se restringe a fórmulas Booleanas en 3-FNC [22]( un problema es PSPACE-completo si pertenece a la clase PSPACE y todo problema en la clase PSPACE se reduce polinomialmente a éste, y un problema pertenece a la clase PSPACE si es soluble en espacio polinomial por una máquina de Turing no determinista). Si $\mathcal{C}$ es un conjunto finito de restricciones, una **expresión** $\mathcal{C}$

**cuantificada (con constantes)** es una expresión de la forma $Q_1x_1Q_2x_2\cdots Q_nx_nS(x_1,\cdots,x_n)$, donde S es un conjunto de aplicaciones restricción de $\mathcal{C}$ (con constantes), y $Q_i \in \{\forall, \exists\}$ para todo i.

La restricción análoga para QBF se define de la siguiente manera:

i) **QSAT($\mathcal{C}$)** es el problema de decidir si una expresión $\mathcal{C}$ cuantificada es verdadera.

ii) **QSAT$_c$($\mathcal{C}$)** es el problema de decidir si una expresión $\mathcal{C}$ cuantificada con constantes es verdadera.

El siguiente teorema propuesto por Schaefer [71] sin demostración en 1978 fué demostrado hasta 2001 por Creignou, Khanna, y Sudan [20], y de manera independiente por Dalmau [22].

**Teorema 2** *Sea $\mathcal{C}$ un conjunto finito de restricciones. Si $\mathcal{C}$ es Horn, anti-Horn, biyuntivo o afín, entonces QSAT($\mathcal{C}$) y QSAT$_c$($\mathcal{C}$) estan en $P$; de otra manera, QSAT($\mathcal{C}$) y QSAT$_c$($\mathcal{C}$) son PSPACE-completos.*

El siguiente resultado es un teorema de dicotomía para problemas de conteo de satisfactibilidad generalizada el cual fue demostrado en [19].

**Teorema 3** ***(Dicotomía de Creignou para $\#SAT$)***.
*$\#SAT(S)$ pertenece a FP si $S$ es afín, y es $\#P$-completo en otro caso.*

Por ejemplo, si $S = \{C_1, ..., C_k : k < n,\ C_i(x_1, ..., x_n) = x_i \wedge \sim x_{i+1},\ i = 1, ..., k)\}$, $\#SAT(S)$ pertenece a FP. En efecto, por el teorema 2 basta verificar que $S$ es afín. Para esto, si $s_1, s_2, s_3$ son tales que $C_i(s_i) = 1$ para $i = 1, 2, 3$, entonces $s_i = (*, ..., *, 1, 0, *, ..., *)$, es decir, $s_i$ tiene el valor de 1 en el lugar $i$, 0 en el lugar $i + 1$, y cualquier otro valor en los lugares restantes. De aquí se deduce fácilmente que $C(s_1 \oplus s_2 \oplus s_3) = 1$. De la proposición 1, $\mathcal{F}$ es afín.
A partir de los resultados de dicotomía de Scheafer y Creignou han surgido una serie de trabajos orientados hacia la identificación de la estructura algebraica de subclases de problemas de conteo de satisfacción de restricciones ($\#CSP$) que pueden ser tratables, entre estos destacan los trabajos de Bulatov [13, 14, 15, 16] y Jeavons [54, 55, 56].
Recientemente Bulatov [13] hace un estudio sistemático de subclases restingidas de $\#CSP$ en donde establece un criterio para hacer la distinción entre las subclases restringidas que son tratables y las que no lo son. Demuestra que la complejidad de cualquier subclase de la clase de $\#CSP$ sobre un dominio finito se puede deducir a través de un conjunto de operaciones llamadas polimorfismos. Este método algebraico le permite identificar una propiedad común de toda restricción en $\#CSP$: demuestra que toda $\#CSP$-clase que no tiene polimorfismo de Mal'tsev (operaciones ternarias m(x,y,z) tales que m(x,y,y)=m(y,y,x)=x) es $\#P$-completo. Bulatov conjetura un posible criterio para $\#CSP$, el cual demuestra para casos particulares sobre restricciones sobre dominios de ternas. La ventaja de este método es que proporciona un nuevo tipo de algoritmos de conteo que utilizan resultados algebraicos y métodos combinatorios.

**Conjetura 1** ***(Conjetura de dicotomía de Bulatov)***. *Un lenguaje de restricción $\Gamma$ es $\#$-tratable si y sólo si $\Gamma$ es invariante bajo una operación Mal'tsev. De otra manera $\Gamma$ es $\#P$-completo.*

Un **lenguaje de restricción** $\Gamma$ sobre un conjunto A es un conjunto finito de predicados sobre A. $\Gamma$ es **$\#$-tratable** si para cualquier subconjunto finito $\Gamma'$ de $\Gamma$, el problema $\#CSP(\Gamma')$ es soluble en tiempo polinomial. $\Gamma$ es llamado $\#$ **P-completo** si $\#CSP(\Gamma')$ es $\#P$-completo para algún subconjunto $\Gamma'$ de $\Gamma$.

Si $A^n$ denota el conjunto de todas la n-tuplas de elementos de $A$, para cualquier operación m-aria $f$, y cualquier colección de tuplas $a_1, \cdots, a_m \in A^n$, donde $a_i = (a_i[1], \cdots, a_i[n])(i = 1 \cdots m)$, se define

$$f(a_1, \cdots, a_m) = (f(a_1[1], \cdots, a_m[1]), \cdots, f(a_1[n], \cdots, a_m[n]))$$

Entonces $f$ **preserva** una relación n-aria $\Re$ (o $\Re$ es **invariante** bajo $f$ o $f$ es un **polimorfismo** de $\Re$) si para cualesquiera $a_1, \cdots, a_m \in \Re$ la tupla $f(a_1, \cdots, a_m) \in \Re$.
Por ejemplo, si $\Gamma$ es como en el ejemplo anterior, esto es $\Gamma = S$, donde $S = \{C_1, ..., C_k : k < n,\ C_i(x_1, ..., x_n) = x_i \wedge \sim x_{i+1},\ i = 1, ..., k)\}$, tenemos que $m : \mathbb{B} \longrightarrow \mathbb{B}$, $m(x, y, z) = x - y + z$ es un polimorfismo de Mal'tsev (donde $\{+, -\}$ son la sustracción y adición módulo 2), puesto que satisface la condición $m(x, y, y) = m(y, y, x) = x$. Entonces, dados $s_1, s_2, s_3$ tales que $C_i(s_i) = 1$ para $i = 1, 2, 3$, se cumple que $C(m(s_1, s_2, s_3)) = 1$, es decir $\Gamma$ es invariante bajo la operación $m$, se sigue de la conjetura de Bulatov que $\Gamma$ es $\#$-tratable. En efecto, la invarianza de $\Gamma$ bajo $m$ se sigue de la igualdad:

$$(*, ..., 1, 0, ..., *) - (*, ..., 1, 0, ..., *) + (*, ..., 1, 0, ..., *) = (*, ..., 1, 0, ..., *).$$

## 6 Razonamiento Proposicional

Dentro del razonamiento proposicional es de interés los problemas que son tratados a través de técnicas de razonamiento aproximado, como son: el cómputo del grado de creencia (base del razonamiento aproximado [69]), revisión y actualización de bases de conocimiento y problemas de satisfacción de restricciones. Otra línea que no abordaremos aquí trata el razonamiento aproximado. Roth demuestra que cada uno de los casos anteriores de inferencia es equivalente a resolver el problema #SAT [69]. A continuación presentamos una descripción de estas técnicas.

### 6.1 Cómputo del Grado de Creencia

En inteligencia artificial surgen varios problemas donde se considera una base de conocimiento inicial que podría ser una teoría proposicional $\Sigma$ y se supone que nos gustaría asignar un grado de creencia a una proposición cualquiera $\alpha$. Por ejemplo, las acciones que un agente toma en un medio ambiente cuando se le presenta incertidumbre, pueden depender fuertemente del grado de creencia que utilice para cuantificar dicha incertidumbre.
En general, para computar el grado de creencia se sigue el siguiente principio: asignar el mismo grado de creencia a todas las "situaciones básicas" consistentes con la base de conocimiento, y calcular la fracción de aquellas que son consistentes con la pregunta (query). Todos los modelos posibles de la teoría tienen igual peso y nos interesa la complejidad computacional del cómputo del grado de creencia de una fórmula proposicional, esto es, la parte de modelos que son consistentes con una pregunta (query) proposicional.
Se tienen trabajos orientados a cómo aplicar este principio y determinar qué son las situaciones básicas [53, 30, 32, 31]. En particular nos interesa el cómputo del grado de creencia en un caso más restrictivo, en el cual la base de conocimiento consiste de una teoría proposicional y no contiene información estadística. Las dificultades computacionales aún en este caso restrictivo son altas, Grove [45] trata la versión de primer orden de este problema donde demuestra que casi todos los problemas que pudieran requerir una respuesta son altamente indecidibles.
Dada una teoría proposicional $\Sigma$ sobre $n$ variables, la probabilidad de que $\Sigma$ sea satisfactible, $P_\Sigma$, es com-

putada con la distribución uniforme sobre el conjunto $\mathbb{B}^n$ de todas las posibles asignaciones de $\Sigma$.

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

donde $M(\Sigma)$ denota el conjunto de todas las asignaciones que satisfacen $\Sigma$, $\mid M(\Sigma) \mid$ denota su cardinalidad y $T$ la constante $Verdadero$.
Dada una teoría proposicional $\Sigma$ y una afirmación proposicional $\alpha$, **el grado de creencia en** $\alpha$ es la probabilidad condicional de $\alpha$ con respecto a $\Sigma$, $P_{\alpha|\Sigma}$, esto es, la parte de asignaciones de satisfactibilidad que satisfacen $\alpha$:

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

Los siguientes resultados sobre la complejidad el el cómputo de $P_\Sigma$ se deben a Roth.

**Proposición 2** *La complejidad de cómputo del grado de creencia en una afirmación proposicional con respecto a una teoría proposicional, esta polinomialmente relacionada con la complejidad de cómputo del número de modelos de una afirmación proposicional.*

**Teorema 4** *El problema de computar el grado de creencia en una afirmación proposicional con respecto a una teoría proposicional, es $\#P - completo$.*

**Redes de Creencia Bayesianas**. Una red de creencia Bayesiana consiste de una estructura de un grafo dirigido acíclico (DAG) aumentada por un conjunto de probabilidades, esto es, una terna (V,E,P) donde:

- V es el conjuto de nodos (que representan variables aleatorias),
- E es el conjunto de aristas ( que representan la existencia de influencia casual entre las variables ligadas),
- P es el conjunto de probabilidades (que consiste de funciones de probabilidad $P(X_i = x_i)$ previas para cada nodo fuente $X_i$, y , funciones de probabilidad condicional $\{P(X_i \mid Y_{ij})\}_{Y_{ij} \in p_{X_i}}$ para cada nodo $X_i$ con un conjunto $p_{X_i}$ de predecesores directos.

Observemos que no toda distribución de probabilidad puede ser representada por una red de creencias Bayesiana, sin embargo dado un DAG se pueden especificar consistentemente las probabilidades condicionales (asegurandose de que para todo nodo $X_i$, $\Sigma_{x_i} P(X_i = x_i \mid p_{X_i}) = 1$). Para un análisis de complejidad se toma como parámetro $n$, el número de nodos de la red de creencia. Notemos que la tabla de probabilidades condicionales asociadas una red de creencias pueden ser de tamaño exponencial en n, por lo que, en general se tratan de evitar estos casos considerando representaciones concisas de la tabla, polinomiales en el número de nodos de la red, entonces podemos tomar en cuenta para nuestra medida de complejidad el total de nodos de la red, incluyendo la tabla de probabilidades condicionales.
El **problema de inferencia** usando redes de creencia consiste en calcular la probabilidad posterior $P(S_1 \mid S_2)$ donde $S_1$ ($S_2$) es una variable o una conjunción de variables . La forma más restrictiva para inferencia probabilística es la determinación de $P(Y = T)$ para alguna variable proposicional $Y$. Cooper demuestra en [**?**] que este problema es NP-difícil, de donde se infiere que no podemos esperar que se desarrollen algoritmos de propósito general para inferencia probabilística de complejidad en tiempo polinomial. Roth [69] demuestra con el siguiente teorema que esta situación no mejora, aún si tratamos de construir **algoritmos de aproximación** para inferencia probabilística.

**Teorema 5** *El problema de computar la probabilidad de que un nodo en una red de creencias Bayesiana sea verdadero es $\#P - completo$. Y para cada $\epsilon > 0$ fijo, aproximar esta probabilidad dentro de $2^{n^{1-\epsilon}}$ es $NP$-difícil (n es el tamaño de la red).*

### 6.2 Revisión-Actualización de Bases de Conocimiento

Recientemente se han publicado nuevos resultados sobre la revisión y actualización de bases de conocimiento [31, 34, 35, 36]. La mayoría de éstos, se basa en diversos métodos para la revisión y actualización de bases de conocimiento como: los "Mundos Posibles" de Ginsberg [43], los estudios de Nebel [64, 65], el Producto-Cruz [39], el WIDTIO [43, 39], la revisión de Dalal [37, 39], la investigación por Forbus [37], el método de Borgida [39], "Modelos posibles" de Winslett [37], método de Satoh [37], y la investigación de Weber [39]. Estos métodos se apegan al principio de *mínimalidad de cambio*, este principio establece que la base de conocimiento deberá cambiar lo mínimo posible cuando se incorpore nueva información. Cada método es adecuado para aplicaciones particulares, de hecho, se tiene la creencia de que no existe un método general que se adecue a cualquier circunstancia [34]. La complejidad computacional de las diferentes propuestas fue presentada como un problema importante en [58, 65]. A pesar de que se sabe que la fórmula de implicación es intratable para cada método, la complejidad precisa de este problema sigue siendo un problema abierto. Más aún, no es claro bajo que restricciones el problema de revisión-actualización es tratable [34]. Presentamos a continuación el siguiente método que en principio, corresponde a la orientación de nuestro proyecto.

Dada una base de conocimiento $\Sigma$, nos interesa revisar $\Sigma$ dada nueva información expresada por una fórmula $F$, esta operación se denota por $\Sigma^*_M F$, donde $M$ es el método empleado para realizar la revisión.

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

$$W(\Sigma, F) = \{\Sigma' \subseteq \Sigma :\sim (\Sigma' \cup F \vdash \bot) \;\&\; \sim \exists \gamma : \Sigma' \subset \gamma \subseteq \Sigma \;\&\; \sim (\gamma \cup F \vdash \bot)\}$$

Donde la notación $F_1 \vdash F_2$, significa que $F_2$ es verdadera para todo modelo de $F_1$ y $\bot$ es la fórmula distinguida "Falsa".
$W(\Sigma, F)$ contiene a todos los subconjuntos plausibles que podemos retener cuando insertamos $F$ [61]. Una de las operaciones más comunes de revisión de cambio de fórmula base es el método de Ginsberg donde:

$$\Sigma^*_G F = \{\Sigma' \cup F : \Sigma' \in W(\Sigma, F)\}. \tag{1}$$

### 6.3 Problemas de Satisfacción de Restricciones

. En general, un problema de satisfacción de restricciones (CSP) se puede plantear de la siguiente forma:

1) dados un conjunto de $n$ variables $x_1, x_2, ..., x_n$ que toman valores en los dominios $D_1, D_2, ..., D_n$ respectivamente y,
2) una colección $S = \{C_1, C_2, ..., C_m\}$ de subconjuntos de $D_1 \times D_2 \times \cdots \times D_n$, llamados $restricciones$.

Se quiere hallar el conjunto

$$SAT(S) = \left\{ v \in D_1 \times D_2 \times \cdots \times D_n : v \in \bigcap_{i=1}^{m} C_i \right\} \tag{2}$$

esto es, el conjunto de todos los valores $v$ que se pueden asignar a la tupla $(x_1, ..., x_n)$ tal que se satisfacen todas las restricciones $C_i$.
Por ejemplo, dada una fórmula $F$ en FNC, cuyas cláusulas son $F_1, ..., F_m$, $F$ se puede ver como un CSP. En efecto, el conjunto dado en 3.4.2 es el conjunto de asignaciones que satisfacen la fórmula $F$, donde $D_i = \mathbb{B}$ (i=1,...,n) y cada conjunto $C_i$ (i=1,...,m) consiste de todas las n-tuplas de valores Booleanos a las variables $x_1, x_2, ..., x_n$ que hacen verdadera la cláusula $F_i$.
El problema de hallar una solución a un CSP (un elemento del conjunto 3.4.2) es un problema difícil en general. Actualmente, se han propuesto diversas heurísticas para dar soluciones aproximadas [4, 38, 48]. Centrando, nuestro interés en los problemas CSP desde el punto de vista de conteo y de cómputo aproximado tenemos los siguientes resultados propuestos por Roth, que establece que el cómputo del número de soluciones a un problema CSP y la aproximación de éste, son problemas difíciles.

**Teorema 6** *El problema de calcular el número de soluciones de un CSP es $\#P$-completo. Y para todo $\epsilon > 0$ fijo, aproximar este número es NP-difícil.*

Existen varios trabajos que proponen heurísticas que trabajan sobre subproblemas simplificados para estimar el número de soluciones de un problema CSP, pero cuando este problema no tiene una estructura trivial el número estimado no es indicativo del número de soluciones del problema original [69, 70].

**Corolario 1** *Usar heurísticas de conteo para problemas CSP es computacionalmente intratable.*

## 7 Resultados en $\#SAT$

### 7.1 Un algoritmo en tiempo lineal para $(\#2\mu, 2)SAT$

Como ya hemos mencionado en 3.1, existen algunos algoritmos eficientes propuestos para resolver $\#SAT$, para fórmulas en $(2\mu, 2) - FC$ [69, 70]. Aunque estos algoritmos involucran tiempo polinomial, no muestran garantías sobre la complejidad en tiempo lineal, además de no considerar cláusulas unitarias. El algoritmo que presentamos aquí extiende el rango de aplicaciones del algoritmo presentado en [26], y considera fórmulas del tipo $(\leq 2, 2\mu) - FC$, manteniendo a la vez la complejidad en tiempo lineal. Una fórmula $(\leq 2, 2\mu) - FC$ es una $(\leq 2)$-forma conjuntiva en la que cada variable aparece a lo más dos veces.

**Notación y preliminares**. Sea $X = \{x_1, ..., x_n\}$ un conjunto de $n$ variables Booleanas. Una **literal** $l$ es un elemento de un conjunto complementario $\{x, \sim x\}$, para alguna variable $x \in X$. Con $\nu(l)$, denotamos la variable involucrada en la literal $l$. Para cada $x \in X$, $x^0 = \sim x$, y $x^1 = x$. Una **cláusula** es una disyunción de literales. Una cláusula es una **tautología**, si esta tiene un par de literales complementarias.
Para una fórmula Booleana $\Sigma$, $\nu(\Sigma) = \{x \in X : x \text{ aparece en } \Sigma\}$, y $Lit(\Sigma) = \{l : \nu(l) \in \nu(\Sigma)$. Una **forma conjuntiva** FC, es una conjunción de cláusulas. Una **asignación** $s$ para $\Sigma$, es una función $s : \nu(\Sigma) \longrightarrow \mathbb{B}$, $(\mathbb{B} = \{0, 1\})$. Una asignación también puede ser considerada como un conjunto de literales donde no hay pares complementarios de literales. Una asignación $s$ **satisface** la cláusula $c$, si $c \cap s \neq \emptyset$, de otra manera decimos que $c$ se **contradice** o se **falsifica** por $s$. Una FC $\Sigma$, se **satisface** por la asignación $s$, si y sólo si

toda cláusula de $\Sigma$ se satisface por $s$. La fórmula $\Sigma$ se **contradice** por $s$ si alguna cláusula de $\Sigma$ se contradice por $s$. Un **modelo** de $\Sigma$ es una asignación sobre $\nu(\Sigma)$ que satisface $\Sigma$. Con $M(\Sigma)$ denotamos el conjunto de modelos que tiene $\Sigma$ sobre $\nu(\Sigma)$. La cardinalidad de $M(\Sigma)$, se denota por $\#SAT(\Sigma)$, esto es: $\#SAT(\Sigma) =\mid M(\Sigma) \mid$.

**Resolviendo $(\#2\mu, \leq 2)$-SAT usando componentes conexas**. Dada una fórmula $\Sigma$ del tipo $(2\mu, 2) - FC$, en [26] fue presentado un algoritmo para $\#SAT$, la extensión de este algoritmo para que incluya fórmulas en 2-FC no estrictas no es trivial, esto se debe a que se tienen que considerar equivalencias lógicas que involucran cláusulas unitarias que no siempre conservan reducciones parsimoniosas. Por ejemplo, las fórmulas $\{(x)\}$ y $\{(x) \wedge (x \vee y)\}$ son lógicamente equivalentes, sin embargo, las cardinalidades de los conjuntos de asignaciones son diferentes y también el número de modelos de cada fórmula.
Dada $P = \{\Sigma_1, ..., \Sigma_r\}$ una partición de una fórmula Booleana $\Sigma$, decimos que $P$ es una partición en componentes conexas de $\Sigma$, si $\{\nu(\Sigma_1), ..., \nu(\Sigma_r)\}$ es una partición en componentes conexas de $\Sigma$, entonces:

$$\#SAT(\Sigma) = \prod_{i=1}^{r} \#SAT(\Sigma_i) \tag{3}$$

El grafo no dirigido inducido por $\Sigma$, es denotado por $G_\Sigma = (V, E)$, y se construye de la siguiente manera: para cada cláusula $c$ en $\Sigma$ tenemos un nodo en $V$ y definimos una arista $(c, c') \in E$ si y sólo si las dos cláusulas $c$ y $c'$ tienen una variable en común. Ver figura 5.1.1.

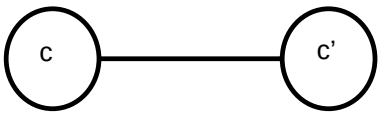


Fig. 5.1.1. Arista entre cláusulas con una variable común.

El dual correspondiente a $G_\Sigma$ puede ser representado como un grafo cuyos nodos corresponden a variables en $\nu(\Sigma)$ y se impone una arista entre cualquier par de variables que aparezcan en la misma cláusula. Ver figura 5.1.2.

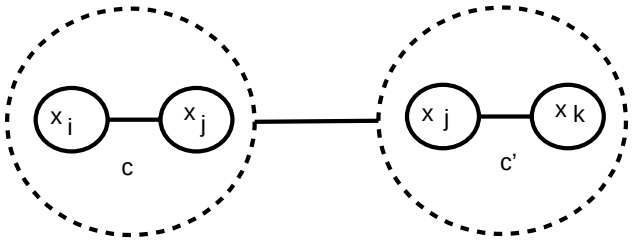


Fig. 5.1.2. Arista entre variables en la misma cláusula.

El grafo dual en [7] es llamado **grafo fórmula** y en [5] se le denomina **grafo fundamental**.
Sea $\{\Sigma_1, ..., \Sigma_r\}$ una partición de $\Sigma$ y $\{G_{\Sigma_1}, ..., G_{\Sigma_r}\}$, los subgrafos inducidos respectivamente, entonces éste último conjunto conforma las diferentes componentes conexas de $G_\Sigma$. Para una fórmula $\Sigma$ del tipo $(\leq 2\mu, 2) - FC$, $G_\Sigma$ tiene las siguientes propiedades:

- El grado máximo de un nodo de $G_\Sigma$ es dos (el grado de un nodo es el número de aristas que inciden en él).

- Dada una cláusula $c \in \Sigma$ la componente conexa de $c$ puede ser obtenida en tiempo lineal [47].
- Las componentes de $G_\Sigma$($\Sigma$ una $(\leq 2\mu, 2) - FC$), son nodos aislados o cadenas lineales maximales o ciclos.
- La transformación de $\Sigma$ a $G_\Sigma$ es una reducción parsimoniosa, como también lo es, la transformación inversa.

**Conteo del número de modelos de cada componente conexa.** Para calcular el número de modelos de una fórmula $\Sigma$ en $(\leq 2\mu, 2) - FC$ , de la ecuación 5.1.1, obtenemos que es suficiente con calcular el número de modelos de cada subgrafo de $G_\Sigma$, que corresponda a una componente conexa. Supongamos que $H$ es una componente conexa de $G_\Sigma$ y que $\Sigma_H$ es su correspondiente subfórmula asociada. A continuación mostramos las diferentes formas de calcular $\#SAT(\Sigma_H)$ de acuerdo al tipo del subgrafo $H$.

**i) H consiste justamente de un nodo**. En este caso tenemos que $\Sigma_H = \{c\}$, entonces:

$$\#SAT(\Sigma_H) = \begin{cases} 1 & \text{si c es una cláusula unitaria,} \\ 3 & \text{si c es cláusula binaria.} \end{cases}$$

**ii) H tiene dos nodos y una arista, y $\Sigma_H$ tiene cláusulas unitarias**. Con estas condiciones se llega a que $\Sigma_H$ tiene un par de cláusulas contradictorias, por lo tanto, $\#SAT(\Sigma_H) = 0$.
**iii) H es un ciclo con dos nodos y dos aristas**. En este caso $\Sigma_H = \{c, c'\}$ donde $\nu(c) = \nu(c')$ y $c$, $c'$ son cláusulas binarias, entonces $\#SAT(\Sigma_H) = 2$, ya que sólo dos asignaciones sobre $\nu(c)$ falsifican $c \wedge c'$.
**iv) H es una cadena lineal**. Para esto supongamos que $H = (V, E)$, donde $\mid V \mid = \mid \{c_1, ..., c_m\} \mid = m$, donde $\mid \nu(c_i) \cap \nu(c_{i+1}) \mid$, $i = 1, ..., (m-1)$. Sin pérdida de generalidad podemos suponer que:

$$\Sigma_H = \{c_1, \{y_1^{\epsilon_2}, y_2^{\delta_2}\}, \{y_2^{\epsilon_3}, y_3^{\delta_3}\}, ..., \{y_{m-2}^{\epsilon_{m-1}}, y_{m-1}^{\delta_{m-1}}\}, c_m\},$$

donde $\delta_i, \epsilon_i \in \mathbb{B}$, $y_1 \in \nu(c_1)$ y $y_{m-1} \in \nu(c_m)$. Entonces

$$\#SAT(\Sigma_H) = \begin{cases} \alpha_{m-1} & \text{si } \epsilon_m = 1 \text{ y } c_m \text{ es una cláusula unitaria,} \\ \beta_{m-1} & \text{si } \epsilon_m = 0 \text{ y } c_m \text{ es cláusula unitaria.} \\ \alpha_m + \beta_m & \text{en cualquier otro caso} \end{cases}$$

donde $\alpha_m$ , $\beta_m$, $\alpha_{m-1}$, y $\beta_{m-1}$ se calculan recursivamente de la siguiente manera:

$$(\alpha_i, \beta_i) = \begin{cases} (\beta_{i-1}, \alpha_{i-1} + \beta_{i-1}) & \text{si } (\epsilon_i, \delta_i) = (0, 0), \\ (\alpha_{i-1} + \beta_{i-1}, \beta_{i-1}) & \text{si } (\epsilon_i, \delta_i) = (0, 1), \\ (\alpha_{i-1}, \alpha_{i-1} + \beta_{i-1}) & \text{si } (\epsilon_i, \delta_i) = (1, 0), \\ (\alpha_{i-1} + \beta_{i-1}, \alpha_{i-1}) & \text{si } (\epsilon_i, \delta_i) = (1, 1). \end{cases} \tag{4}$$

donde $i \in \{2, ..., m-1\}$ si $c_1$ es una cláusula binaria y $i \in \{3, ..., m-1\}$ si $c_1$ es una cláusula unitaria. Los casos base son:

$$(\alpha_1, \beta_1) = \begin{cases} (1, 2) & \text{si } \delta_1 = 0, \\ (2, 1) & \text{si } \delta_1 = 1 \end{cases}$$

y

$$(\alpha_2, \beta_2) = \begin{cases} (1,1) & \text{si } \epsilon_1 = \epsilon_2, \\ (1,0) & \text{si } \delta_1 = 1, \\ (0,1) & \text{de otra manera} \end{cases}$$

respectivamente.

**v) H es un ciclo con m nodos**. En este caso todas las cláusulas de $\Sigma_H$ son binarias y bajo un orden adecuado de estas podemos escribir:

$\Sigma_H = \{\{y_{i-1}^{\epsilon_i}, y_i^{\delta_i}\}\}_{i=1}^m$, donde $y_0 = y_m$ y $\delta_i, \epsilon_i \in \mathbb{B}$.

La primera cláusula $\{y_m^{\epsilon_1}, y_1^{\delta_1}\}$ tiene tres asignaciones que la satisfacen, $s_1 = \{y_m^{1-\epsilon_1}, y_1^{\delta_1}\}$, $s_2 = \{y_m^{\epsilon_1}, y_1^{1-\delta_1}\}$, y $s_1 = \{y_m^{\epsilon_1}, y_1^{\delta_1}\}$. De acuerdo al signo de $y_1$ definimos:

$$(\alpha_1, \beta_1)(s_1) = \begin{cases} (1,0) & \text{si } \delta_1 = 1, \\ (0,1) & \text{de otra manera,} \end{cases}$$

$$(\alpha_1, \beta_1)(s_2) = \begin{cases} (0,1) & \text{si } \delta_1 = 1, \\ (1,0) & \text{de otra manera,} \end{cases}$$

$$(\alpha_1, \beta_1)(s_3) = \begin{cases} (1,0) & \text{si } \delta_1 = 1, \\ (0,1) & \text{de otra manera.} \end{cases}$$

Entonces tenemos que $\#SAT(\Sigma_H) = t(s_1) + t(s_2) + t(s_3)$, donde:

$$t(s_\lambda) = \begin{cases} \alpha_m & \text{si } y_m \in s_\lambda, \\ (0,1) & \text{si } \bar{y}_m \in s_\lambda. \end{cases}$$

y $(\alpha_i, \beta_i)(s_\lambda)$ se calcula para $\lambda = 1, 2, 3$ usando la fórmula de recurrencia 5.1.2.

**Un algoritmo para resolver #2,2-SAT**. Notemos primero que para cada clausula $c \in \Sigma$, la componente conexa de $c$ puede ser obtenida en tiempo lineal con respecto al tamaño de $\Sigma$ , siempre y cuando existan apuntadores a cada una de las cláusulas que indiquen en qué cláusula y con qué signo aparece en la fórmula [47]. Para calcular $\#SAT(\Sigma)$, a veces es necesario particionar primero la fórmula $\Sigma$, lo que nos conduce a escanear la fórmula $\Sigma$ varias veces, una forma de evitar esto, es auxiliarnos de una tabla de apuntadores como los que acabamos de describir. Con esta tabla y siguiendo los procedimientos de la sección anterior podemos calcular $\#SAT(\Sigma)$ siguiendo el algoritmo:

1) Primero, observemos en la tabla las variables que ocurren en la fórmula una sola vez. Las cláusulas donde estas variables ocurren son nodos aislados o son el primer nodo de una cadena lineal. Sea $c$ una de estas cláusulas, si $c$ es unitaria o si $c$ es binaria y ambas variables aparecen una vez, entonces $c$ representa un nodo aislado, y así podemos proceder como en (i), de otra manera aplicamos la opción (iv).

2) Después de esto, procesamos las cláusulas unitarias. Si existen dos cláusulas unitarias contradictorias aplicamos (iii), o procedemos a procesar una cadena lineal por la opción (iv).

3) Finalmente, tenemos en las cláusulas restantes en la tabla solamente subfórmulas que representan ciclos, así que podemos aplicar (v).

**Comentario**. Este procedimiento calcula $\#SAT(\Sigma)$ para $\Sigma$ en $(\leq 2, 2\mu) - FC$ con complejidad en tiempo lineal ya que se basa en la aplicación de (1) cada vez que cada nodo de una componente conexa es visitado. Existen otros procedimientos para calcular $\#SAT(\Sigma)$ [69, 70], pero estos tienen el inconveniente de que no garantizan la complejidad en tiempo lineal, además de que al contar los modelos de $\Sigma$, no distinguen entre los modelos en los que una variable $x$ toma el valor de 1 de aquellos en el que la misma variable $x$ toma el valor de 0. En nuestro procedimiento esta situación se hace explícita a través del par $(\alpha, \beta)$. Como veremos en las secciones siguientes esta distinción es esencial para contar modelos de fórmulas a las que se les añaden nuevas fórmulas.

### 7.2 Nuevas clases polinomiales de 2-FC para $\#SAT$

De acuerdo a la jerarquía $\#SAT$ propuesta por Russ, hemos identificado nuevas clases polinomiales de fórmulas Booleanas en 2-FNC para $\#SAT$. Estos resultados se encuentran por arriba de $\#2\mu - 2SAT$ y por abajo de $\#3\mu - 2SAT$. Como ya se ha mencionado en 3.1, para $\#2\mu - 2SAT$ el conteo exacto es posible hacerlo en tiempo polinomial, sin embargo para $\#3\mu - 2SAT$ el conteo exacto en tiempo polinomial permanece como problema abierto(ver tabla 3).
A continuación damos una descripción de los conceptos y resultados que nos conducen a la identificación de estas nuevas clases polinomiales.

**Grafo inducido por una fórmula en 2-FC**. Dada $\Sigma$ una 2-FC, como la disjunción es conmutativa, sin pérdida de generalidad podemos suponer que $\Sigma$ se encuentra ordenado de la siguiente manera: si $(l_i, l_j) \in \Sigma$, entonces $i < j$, esto es; $\Sigma$ puede ser interpretada como un conjunto de pares ordenados de acuerdo al índice de cada literal y donde cada par es una cláusula y cada literal es una variable o la negación de esta. Definimos $G_\Sigma$, el grafo inducido por $\Sigma$ de la siguiente manera:

1) el conjunto $V$ de vértices de $G_\Sigma$ es el conjunto de variables de $\Sigma$, $\nu(\Sigma)$,

2) el conjunto de aristas E de $G_\Sigma$ es el conjunto de cláusulas de $\Sigma$ (observemos que dada la arista $(l_i, l_j) \in \Sigma$ se puede interpretar como un el arco etiquetado: $\nu(l_i) \xrightarrow[sg(l_i)sg(l_j)]{} \nu(l_j)$, donde $sg(l)$ denota el signo de la literal $l$).

La figura 5.1.3 muestra un ejemplo del grafo asociado a la fórmula $\Sigma = \{(x_1, \sim x_2), (x_2, x_3), (\sim x_3, \sim x_4), (x_1, \sim x_3), (x_2, x_4))\}$.

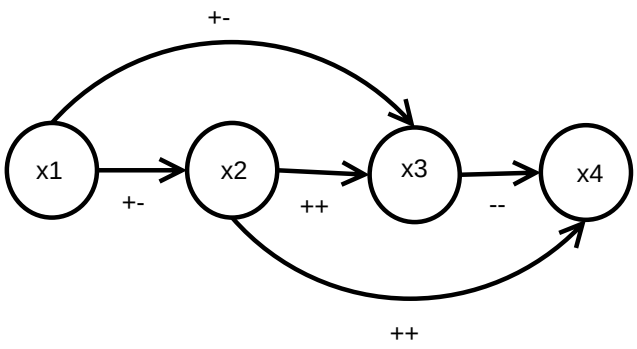


Fig. 5.1.3. Grafo $G_\Sigma$.

**Operadores Matriciales**. Introducimos aquí un grupo de operadores matriciales que opera sobre el espacio $M_{2\times 2}$ de matrices $2 \times 2$, el cual clasificamos de la siguiente manera:

**Operadores Cadena**. Este grupo consiste de los operadores definidos por:

$$T_{++} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, T_{+-} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, T_{-+} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, T_{--} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

**Operadores Proyección**. Los operadores proyección $\pi_+$ y $\pi_-$ son:

$$\pi_+ \begin{pmatrix} a & 0 \\ b & 0 \end{pmatrix} = \begin{pmatrix} a & a \\ b & 0 \end{pmatrix}, \pi_- \begin{pmatrix} a & 0 \\ b & 0 \end{pmatrix} = \begin{pmatrix} a & 0 \\ b & b \end{pmatrix}.$$

**Operadores Diagonal**. Los símbolos $d_+$ y $d_-$ denotan:

$$d_+ \begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} a & 0 \\ d & 0 \end{pmatrix}, d_- \begin{pmatrix} a & c \\ b & d \end{pmatrix} = \begin{pmatrix} c & 0 \\ b & 0 \end{pmatrix}.$$

Observemos que las potencias de los operadores cadena tienen en sus entradas los números de Fibonacci, por ejemplo:

$$T_{++}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

Donde $F_0 = 0, F_1 = 1, F_{n+1} = F_n + F_{n-1}$.
Los operadores proyección $\pi_+$ y $\pi_-$ operan sobre el subespacio de $M_{2\times 2}$ de las matrices cuya segunda columna es el vector nulo (0,0).

**Operador inducido por** $G_\Sigma$. Si $\Sigma$ es una cadena, podemos asociar a $G_\Sigma$ el operador $O_\Sigma$ definido a continuación.
Supongamos $\Sigma = \{(l_i, l_{i+1}) : i = 1, ..., n\}$, para cada cláusula $(l_i, l_{i+1})$ en $\Sigma$ tenemos un operador $T_{sg(l_i)sg(l_{i+1})}$, para la cadena completa el operador inducido por $G_\Sigma$ se define por

$$O_\Sigma = \prod_{i=1}^{n} T_{sg(l_{n-i})sg(l_{n+1-i})} \tag{5}$$

Por ejemplo, el grafo de la cadena $\Sigma = \{(x_1, \sim x_2), (x_2, x_3), (\sim x_3, \sim x_4)$ tiene asociado el operador $O_\Sigma = T_{--}T_{++}T_{+-}$.

Dada una cadena de símbolos $s = s_1 s_2 \cdots s_m$ de algún alfabeto $\Gamma$, supongamos que $i_1, i_2, \cdots, i_n \in \{1, 2, \cdots, m+1\}, i_k < i_{k+1}, k = 1, \cdots, n$, definimos la función inserción $I_s : \mathbb{N}^n \times \Gamma^n \longrightarrow \Gamma^*$ como sigue:

$I_s(i_1, \cdots, i_n, a_1, \cdots, a_n) = x_1 \mathbf{s_1} x_2 \mathbf{s_2} x_3 \cdots x_m \mathbf{s_m} x_{m+1}$ donde

$$x_p = \begin{cases} a_k & \text{si } p = i_k, k \in \{1, \cdots, n\}, \\ \\ \epsilon & \text{de otra manera.} \end{cases}$$

para todo $p = 1, 2, \cdots, m+1$.

Si $\Sigma$ es una cadena y $(l, l')$ es una cláusula tal que $\nu(\{l, l'\}) \subseteq \nu(\Sigma)$, entonces $\Sigma \cup \{(l, l')\}$ es una cadena con un ciclo y el arco $\nu(l) \overrightarrow{\ \ \ \ } \nu(l')$ es llamado arco ciclo. Decimos que dos arcos:

$e_{ij} = \nu(l_i) \overrightarrow{\ \ \ \ } \nu(l_j)$ y $e_{kt} = \nu(l_k) \overrightarrow{\ \ \ \ } \nu(l_t)$.

no tienen nodos comunes si $i < k$ o $t < i$.

Consideremos $G_\Sigma = (V, E)$ una cadena lineal, donde $V = \{x_1, x_2, \cdots, x_m\}$. Definimos el operador inducido por $G_{\Sigma'}$, donde $G_{\Sigma'} = (V', E')$, $V' = V$, $E' = E \cup E''$, $E''$ un conjunto de arcos ciclo sin nodos comunes a pares, como sigue:
Para cada arco ciclo

$e_{ij} = \nu(l_i) \overrightarrow{\ \ \ \ } \nu(l_j)$

insertamos en la cadena $O_\Sigma$ dada en 5.1.1, los siguientes símbolos:

$d_{sg(l_i)sg(l_j)}$ en la posición $i$, si $\nu(l_i)$ es el extremo inicial de un arco ciclo,
$\pi_{sg(l_i)sg(l_j)}$ en la posición $j$, si $\nu(l_j)$ es el extremo final de un arco ciclo.

Por lo consiguiente el operador inducido por $G_{\Sigma'}$ es

$$O_{\Sigma'} = I_s(i_1, j_1, \cdots, i_p, j_p, \pi_{sg(l_{i_1})}, d_{sg(l_{j_1})}, \cdots, \pi_{sg(l_{ip})}, d_{sg(l_{jp})}) \tag{6}$$

Ahora estamos en condiciones de enunciar el siguiente resultado.

**Proposición 3** *Sea $\Sigma$ una fórmula en 2-FC tal que $\Sigma = \Sigma' \cup \Sigma''$, donde el grafo dirigido $G_{\Sigma'}$es una cadena, $\Sigma'$ es una 2-FC con m cláusulas donde $\nu(\Sigma'') \subseteq \nu(\Sigma')$ y $G_{\Sigma'} \cup G_{\Sigma''}$ es un grafo dirigido con m ciclos tales*

*que no tienen nodos comunes entre cualquier par de ciclos. Entonces*

$$\#SAT(\Sigma) = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] \tag{7}$$

*Donde* $Sum : M_{2\times 2} \longrightarrow \mathbb{N}$ *es la función definida por*

$$Sum[\begin{pmatrix} a & c \\ b & d \end{pmatrix}] = a + b, \tag{8}$$

$O_\Sigma$ es el operador inducido por el grafo $G_\Sigma$.

Esta proposición establece que podemos computar eficientemente $\#SAT$ para cadenas con ciclos sin nodos comunes utilizando los operadores que hemos identificado.

Para la demostración de este resultado consideramos primero los siguientes lemas.

**Lema 1** *Sea* $\Sigma$ *una fórmula en 2-FC, tal que el grafo* $G_\Sigma$ *es una cadena. Entonces*

$$\#SAT(\Sigma) = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] \tag{9}$$

*donde* $O_\Sigma$ *esta dado en (5) y* $Sum$ *es como en la proposición 3.*

**Demostración**. Para esto aplicamos inducción sobre el número de cláusulas de $\Sigma$. Supongamos que $\Sigma = \{(l_1, l_2)\}$, de (5):

$$O_\Sigma = T_{sg(l_1)sg(l_2)}, O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \in \{\begin{pmatrix} 2 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 2 & 0 \end{pmatrix}\}$$

entonces, (9) se cumple trivialmente. Ahora supongamos que tenemos $\Sigma = \Sigma' \cup \{l_n, l_{n+1}\}$, donde $l_n \in Lit(\Sigma')$ y $|\Sigma'| = n - 1$, entonces

$$\#SAT(\Sigma') = Sum[O_{\Sigma'} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] = Sum \begin{pmatrix} k & 0 \\ m & 0 \end{pmatrix} = k + m,$$

$k$ es el número de asignaciones que satisfacen $\Sigma'$ donde $\nu(l_n)$ toma el valor de 1, y $m$ es el número de asignaciones que satisfacen $\Sigma'$ donde $\nu(l_n)$ toma el valor de 0. Hay cuatro casos para $sg(l_n)sg(l_{n+1})$:

**Caso ++**. $(l_n, l_{n+1}) = (x_n, x_{n+1})$. Para cada asignación que satisface $\Sigma'$ donde $x_n$ toma el valor de 1 existen dos posibilidades para el valor de $x_{n+1}$ que satisfacen la cláusula $(x_n, x_{n+1})$, y para cada asignación donde $x_n$ toma el valor de 0, existe sólo una posibilidad para el valor de $x_{n+1}$ que satisface $(x_n, x_{n+1})$, entonces hay un total de $2k + m$ asignaciones que satisfacen $\Sigma$. De donde,

$$\#SAT(\Sigma) = 2k+m = Sum[\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} k & 0 \\ m & 0 \end{pmatrix}] = Sum[O_{\Sigma'} T_{++} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}].$$

**Caso -+**. $(l_n, l_{n+1}) = (\sim x_n, x_{n+1})$. Para cada asignación que satisface $\Sigma'$ donde $x_n$ toma el valor de 1, hay sólo una posibilidad para el valor de $x_{n+1}$ que satisface $(x_n, x_{n+1})$, y para cada asignación donde $x_n$

toma el valor 0, hay dos posibilidades para el valor de $x_{n+1}$ que satisfacen $(\sim x_n, x_n + 1)$, entonces hay $k + 2m$ asignaciones que satisfacen $\Sigma$. De aquí

$$\#SAT(\Sigma) = k+2m = Sum[\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} k & 0 \\ m & 0 \end{pmatrix}] = Sum[O_{\Sigma'} T_{-+} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}].$$

Las demostraciones de los casos (+-), (–) son análogas a los casos anteriores. □

**Lema 2** *Sea $\Sigma$ una fórmula en $2-FC$, tal que el grafo dirigido $G_\Sigma$ es un ciclo, esto es $\Sigma = \Sigma' \cup \{(l_1, l_n)\}$, $\Sigma'$ es una cadena, $\nu(l_1), \nu(l_n) \in \nu(\Sigma')$. Entonces*

$$\#SAT(\Sigma) = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] \tag{10}$$

*donde $O_\Sigma = d_{sg(l_n)} O_{\Sigma'} \pi_{sg(l_1)}$, $O_{\Sigma'}$ es como en (5) y $Sum$ como en la proposición 3.*

**Demostración**. Por el lema 1,

$$\#SAT(\Sigma') = Sum[O_{\Sigma'} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] = Sum[\begin{pmatrix} k & 0 \\ m & 0 \end{pmatrix}] = k + m,$$

donde k es el número de asignaciones que satisfacen $\Sigma'$ tal que $\nu(l_n)$ toma el valor de 1, y m es el número de asignaciones que satisfacen $\Sigma'$ tal que $\nu(l_n)$ toma el valor de 0. Se tienen cuatro casos para $sg(l_1)sg(l_n)$:

**Caso ++**. Sea $m'$ el número de asignaciones que satisfacen $\Sigma'$ donde $\nu(l_1)$ toma el valor de 1 y $\nu(l_n)$ toma el valor 0, éstas satisfacen la cláusula $(l_1, l_n)$ y por lo consiguiente satisfacen la fórmula $\Sigma$. El número $k$ de asignaciones que satisfacen $\Sigma'$ donde $\nu(l_n)$ toma el valor de 1, también satisface $\Sigma$. Entonces

$$\#SAT(\Sigma) = Sum[\begin{pmatrix} k & 0 \\ m' & 0 \end{pmatrix}] = m'+k = Sum[d_+ \begin{pmatrix} k & k' \\ m & m' \end{pmatrix}] = Sum[d_+(\begin{pmatrix} k & 0 \\ m & 0 \end{pmatrix} + \begin{pmatrix} 0 & k' \\ 0 & m' \end{pmatrix})] =$$

$$Sum[d_+(O_{\Sigma'} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} + O_{\Sigma'} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix})] = Sum[d_+ O_{\Sigma'} \pi_+ \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}] = Sum[O_\Sigma \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}]$$

donde $k'$ es el número de asignaciones que satisfacen $\Sigma'$ tales que $\nu(l_1)$ toma el valor de 1 y $\nu(l_n)$ también toma el valor de 1. Los casos (+-), (-+) y (–) se demuestran de manera similar. □

**Demostración de la proposición 3** . La demostración se sigue por inducción sobre el número de ciclos, utilizando el lema 2 y la descomposición $\Sigma = \Sigma_1 \cup \Sigma_2 \cup ... \cup \Sigma_n$, donde cada $\Sigma_i$ es o una cadena o un ciclo para todo $i \in \{1, ..., n\}$. □

**Comentario**. A diferencia de los resultados obtenidos en (8.1), la complejidad en tiempo es polinomial, pero si hacemos una combinación de estos resultados con los procedimientos dados en (8.1) obtenemos una clase polinomial más general que incluye fórmulas en $(2, 2\mu) - FC$. Creemos que la identificación de nuevos operadores matriciales nos ayudaría a extender aún más esta clase.

## 8 Resultados en Razonamiento Proposicional

### 8.1 Revisión Eficiente de Bases de Conocimiento

Como se ha visto en (6.2) un problema importante es explorar la complejidad computacional de los métodos de revisión de creencias de bases de conocimiento, aunque todos los métodos que se conocen hasta ahora son intratables en el caso general [34, 58, 61], es relevante encontrar bajo que restricciones algunos métodos pudieran ser tratables. Nosotros hemos presentado bajo la idea principal del Dr. De Ita un nuevo operador de revisión de creencias [27], dado como:

$$\Sigma_I^* F = \{\Sigma' \cup F : I \in SAT(F) \ \& \ \Sigma' \in W(\Sigma, F)\}$$

y,

$$W(\Sigma, F) = \{\Sigma' \subseteq \Sigma :\sim (\Sigma' \cup F \vdash \bot) \ \& \ \sim \exists \gamma : \Sigma' \subset \gamma \subseteq \Sigma \ \& \ \sim (\gamma \cup F \vdash \bot)\}$$

$\Sigma \vdash F$ denota cuando $F$ es verdadero para todo modelo de $\Sigma$. Con el símbolo $\bot$ representamos a la fórmula distinguida "Falsa".

**Comentario**. Este nuevo operador tiene una complejidad exponencial en tiempo sobre la longitud de $F$ pero continua siendo polinomial sobre la longitud de $\Sigma$, el cual se considera en general, en conjunto clausular con mucho más cláusulas que las que tiene $F$.

## References


[1] E. Alba, H. Alfonso, B. Dorronsoro. Advanced Models of Cellular Genetic Algorithms Evaluated on SAT. ACM, GECCO05, Washington, DC, USA. pages 1123-1130 (2005).

[2] J.J. Alferes, J.A. Leite, L.M. Pereira, H. Przymusinska, T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. Jour. Of Logic Programmingc 45, pages 43-70 (2000).

[3] J.J. Alferes, L.M. Pereira. Logic Programming updating - a guided approach, F. Sadri, A. Kakas, editors, Essays in honour of Robert Kowalsky, LNAI, Vol. 2, 2408, (2000).

[4] N. Alon, W. F. Vega, R. Kannan, M. Karpinski. Randodom sampling and approximation of MAX-CSP problems. J. Comput. Syst. Sci. 67(2,) pages 212-243, (2003).

[5] F. Bacchus, S. Dalmao, T. Pitassi. DPLL with caching a new algorithm for $\#SAT$ and Bayesian Inference. ECCC Report No.3,Dep. of CS, U. of Toronto,(2003).

[6] N. Bansal, V. Raman, Upper bounds for MaxSat: Further improved. In Aggarwal and Rangan (eds.): Proceedings of 10th Annual conference on Algorithms and Computation, ISSAC99, volume 1741 of Lecture Notes in Computer Science, pages 247-258, Springer- Verlag, (1999).

[7] R. Bayardo, J. Pehoushek. Counting Models using Connected Components. Proceeding of the seventeenth National Conf. on Artificial Intelligence, pages 157-162 (2000).

[8] E. B*ö*hler, E. Hemaspaandra, S. Reith, H. Vollmer. The Complexity of Boolean Constraint Isomorphism. ACM, pages 1-31, (2004).

[9] E. Bohler, N. Creignou, S. Reith, and H. Vollmer. Playing with boolean blocks, part II:constraint satisfaction problems. ACM SIGACT-Newsletter, 35(1):22-35, (2004).

[10] B. Borchers, J. Furman, A two-phase exact algorithm for MAX-SAT and weighted MAXSAT problems. Journal of Combinatorial Optimization, 2(4): pages 299-306, (1999).

[11] R. Bubley, M.Dyer. Graph orientations with no sink and an approximations for a hard case of $\#SAT$. In: Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. New Orleans, Louisiana, pages 248-257 (1997).

[12] R. Bubley, M.Dyer. Path coupling: A technique for proving rapid mixing in Markov chains. In: 38th Annual Symposium on Foundations of Computer Science. Miami Beach, Florida:IEEE pages 223-231 (1997)

[13] A. Bulatov, V. Dalmau. Towards Dichotomy Theorem for the Counting Constraint Sactifaction Problem. Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science. pages 562-571 (2003)

[14] A. Bulatov, P. Jeavons. Algebraic approach to multisorted constraint. Technical Report PRG-RR-01-L8. Computing Laboratory, University Oxford, UK,(2001)

[15] A. Bulatov, P. Jeavons. Algebraic structures in combinatorial problems. Technical Report MATH-AL-4-2001. Technische universit*ä*t Dresden, Germany (2001)

[16] A. Bulatov, A. Krokhin, P. Jeavons. Contraint satisfaction problems and finite algebras. In proceedings of 27th International Colloquium on Automata, Languages and Programming-ICALAP'00, volume L853 of Lecture Notes in Computer Science, pages 272-282. Springer-Verlag,(2002).

[17] P. Chauhan, E. M. Clarke, D. Kroening. A SATBased Algorithm for Reparameterization in Symbolic Simulation. ACM,(DAC 2004), San Diego, California, USA. pages 524-529. (2004).

[18] SA Cook. The complexity of theorem-proving procedures. In: Conference Record of Third Annual ACM Symposium on Theory of Computing. Shaker Heights, Ohio,3-5 ;pages 151-158.4, 5,127 (1971).

[19] N. Creignou, M. Hermann. Complexity of Generalized Satisfiability Counting Problems. Information and Computation, pages 1-12,(1996).

[20] N. Creignou, S. Khanna, and M. Sudan. Complexity Classifications of Boolean Constraint Satisfaction Problems. SIAM Press, Philadelphia, USA, (2001).

[21] N. Creignouy, M. Hermannz. Complexity of Clausal Constraints Over Chains. Seventh International Workshop on Logic and Computational Complexity LCC, Chicago, (2005).

[22] V. Dalmau. Some Dichotomy Theorems on Constant-free Quantified Boolean Formulas. Technical Report TR-LSI-97-43-R, Universitat Polyt'echnica de Catalunya, (1997).

[23] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schoning. A deterministic $(2-2/(k+1))^n$ algorithm for kSAT based on local search. Theoretical Computer Science, (2002).

[24] M. Davis, H. Putman. A computing procedure for quantification theory. J. ACM 7 , pages 201-215, (1960).

[25] M. Davis, G. Logemann, D. Loveland, A machine program for theorem-proving. Communications of the Association for Computing Machinery,pages 394397, (1962).

[26] G. De Ita, G. Morales, $\#2, 2-SAT$ is solvable in linear-time, CO98 Int. Symp. on Combinatorial Optimization, Université Libre de Bruxelles, (1998).

[27] G. de Ita, C. Guillén, D.Pinto. A Linear-Time Algorithm for the 2 ,2-SAT Problem Using Connected Components. "Workshop on Deduction and Reasoning Techniques" , IX Ibero-American Conference on Artificial Intelligence, pp. 13-18, (2004).

[28] G. De Ita, C. Guillén, A. López. Efficient Revision of Propositional Knowledge Bases. Research on Computing Science, Advances in Artificial Intelligence and Computer Science, vol 14, pp. 69-79, (2005).

[29] M. Dyer,C. Greenhill. Some $\#P$-completeness proofs for colourings and independent sets. Technical Report 97.47, School of Computer Studies, University of Leeds,pages 38,130 (1997).

[30] J.P. Delgrande, A.C. Nayak, M. Pagnucco. Gricean Belief Change. Studia Logica, 79, (2005).

[31] J.P. Delgrande, T. Schaub. Two Approaches to Merging Knowledge Bases. 9th European Conference on Logics in Artificial Intelligence, Lisbon Portugal, (2004).

[32] J.P. Delgrande, A. C. Nayak, M. Pagnucco. Conservative Belief Change. American Association for Artificial Intelligence Conference, San Jose, CA (2004).

[33] M. Dyer,C. Greenhill. On Markov chains for independent sets.(Preprint) http://www.scs.leeds.ac.uk/rand/psfiles/ind-sets.ps.,pages 8,25,83,103-105,109,110,115,130, (1997). Technical Report 97.47, School of Computer Studies, University of Leeds,pages 38,130 (1997).

[34] T. Eiter, G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updates, and Counterfactuals. Artificial Intelligence 57,pages 227-270,(1992).

[35] T. Eiter, M. Fink, G. Sabattini, H. Thompits. Considerations on update of logic programs. Lecture Notes in Artificial Intelligence, Vol.1,(JELIA'00). pages 212-226,(2000).

[36] T. Eiter, K. Makino, Generating all abductive explanations for queries on propositional Horn Theories. INFSYS Reseach Report 1843,pages 03-09,(2003).

[37] R. Fagin, Kuper, J. Ullman, and M.Y. Vardi. Updating Logical Databases. In P.Kanellakis and F.Preparata, editors, Advances in Computing Research, volume 3. JAI Press, pages 1-18 (1986).

[38] T. Feder. Constraint Satisfaction: A Personal Perspective. http://theory.stanford.edu/ tomas/ (2005).

[39] R. Fagi, J.D. Ullman, and M.Y. Vardi. On the Semantics of Updates in Databases. In Procceedings PODS-83, pages 352-365,(1983).

[40] G. Flouris, D. Plexousakis, G. Antoniou. AGM Postulates in Arbitrary Logics: Initial Results and Applications. Technical Report FORTH-ICS/TR-336, (2004).

[41] M. Furer',S. P. Kasiviswanathan. Algorithms for Counting 2-SAT Solutions and Colorings with Applications,Electronic Colloquium on Computational Complexity, Report No. 33 pages 1-17 (2005).

[42] MR. Garey, DS. Johnson. Computers and Intractability: a Guide to the Theory of NP-Completeness. San Francisco, California: W.H.Freeman and Company, pages 1,2,55,27,62. (1994).

[43] M.L.Ginsberg.Conunterfactuals. Artificial Intelligence, 30:35-79,(1986).

[44] J. Gramm, E.A. Hirsch, R. Niedermeier, P. Rossmanith: New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. Preprint, submitted to Elsevier, May,(2001).

[45] A. Grove, J. Y. Halpern, D. Koller. Asymptotic condicional probabilities for first-order logic. in ACM Symp. of the Theory of Computing, number 24, pages 294-305, (1992).

[46] C.Guillén, G. de Ita, A. López. Counting the Number of Models of a 2-CF Using Matrix Operators, Taller de Razonamiento y Deducción Automática del Encuentro Nacional de Computación ENC, (2005).

[47] D. Gusfield, L. Pitt. A bounded approximation for the minimum cost 2-Sat problem. Algorithmica 8, pages 103-117, (1992).

[48] S. Hamissi, M. Babes. A Neural Approach for Solving the Constraint Satisfaction Problem. International Conference on Geometric Modeling and Graphics. IEEE, pages 96- (2003).

[49] P. Hansen, B. Jaumard. Algorithms for the maximum satis.ability problem. Computing, 44: pages 279-303, (1990).

[50] E. Hemaspaandra. Dichotomy Theorems for Alternation-Bounded Quantified Boolean Formulas. CoRR: Computational Complexity. arXiv:CC/0406006 V1. (2004).

[51] E.A. Hirsch. New worst-case upper bounds for SAT. Journal of Automated Reasoning, 24(4):397-420, (2000).

[52] E.A. Hirsch. A new algorithm for MAX-2-SAT. In Proceedings of 17th International Symposium on Theoretical Aspects of Computer Science, vol. 1770, Lecture Notes in Computer Science. Springer-Verlag.pages 65-73,STACS (2000).

[53] A. Hunter, J.P. Delgrande. Iterated Belief Change: A Transition System Approach. 19th International Joint Conference on Artificial Intelligence, Edinburgh, (2005).

[54] P.Jeavons. On the algebraic structure of combinatorial problems. Theoretical Computer Science,200:L85-204,(1998).

[55] P.Jeavons, D. Cohen, and M.Cooper.Constraints, consistency and closure. Artificial Intelligence,101(1-2):251-265,(1998).

[56] P.Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints, Journal of the ACM,44:302-332,(2000).

[57] K. Katayama,T. Koshiishi, H. Narihisa. Reinforcement Learning Agents with Primary Knowledge Designed by Analytic Hierarchy Process, ACM Symposium on Applied Computing, pages 14-21,(2005).

[58] H. Katsuno and A. O. Mendelzon. On the Difference between Updating a Knowledge Base and Revising it. In Proceeding KB-91,pages 387-395,(1991).

[59] H. Kautz, B. Selman. A general framework for knowledge compilation. In Proccedings of the International Workshop on Processing Declarative Knowledge, Kaiserlautern, Germany, (1991).

[60] H. Kautz, B. Selman. Forming concepts for fast inference. In Proceeding of the National Conference on Artificial Intelligence, pages 786-793,(1992)

[61] P. Liberatore, M. Schaerf. The Complexity of Model Checking for Belief Revision and Update. Procc. Thirteenth Nat. Conf. on Artificial Intelligence AAAI,(1996).

[62] M. Luby, E Vigoda. Approximately counting up to four. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing.El Paso,Texas, pages 682-687.17,103,104,105,130,(1997).

[63] A. Mishchenko, R. K. Brayton.SAT-Based Complete Dont-Care Computation for Network Optimization, ACM, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition,(2005).

[64] B. Nebel. A Knowledge Level Analysis of Belief Revision. In Procceding KR-89, pages 301-311, (1989).

[65] B. Nebel. Belief Revision and Default Reasoning:Syntax-Based Approaches. In Procceding KR-91, pages 417-428,(1991).

[66] R. Niedermeier, P. Rossmanith. New upper bounds for maximum satis.ability. Journal of Algorithms, 36:63-88, (2000).

[67] C. H. Papadimitriou. Computational Complexity. Addison Wesley, (1994).

[68] O. Roussel. Another SAT to CSP Conversion. Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pages 558-565, ( 2004)

[69] D. Roth. On te hardness of approximate reasoning. Artificial Intelligence, pages 273-302, (1996).

[70] B.Russ. Randomized Algorithms: Approximation, Generation, and Counting. Springer-Verlag London Berlin Heidelberg. (2001).

[71] T. Schaefer. The complexity of satisfiability problems. In Proceedings of the 10th ACM Symposium on Theory of Computing, pages 216-226, (1978).

[72] S. E. Shimony, C. Domshlak. Complexity of Probabilistic Reasoning in Directed-Path Singly Connected Bayes Networks. Elsevier Science Publishers Ltd. Essex, UK. Artificial Intelligence , Volume 151, Pages: 213-225 , Issue 1-2, (2003)

[73] B. Selman, H. Kautz. Knowledge compilation using Horn approximations. In Proceeding of the National Conference on Artificial Intelligence, pages 904-909, (1991).

[74] H. Sheldon y T. Aytemiz. A Probabilistic Study of 3-Satisfibability. Theory of Computing, pp. 22- 29. (2001).

[75] A. Sinclair. Algorithms for Random Generation and Counting: A Markov Chain Approach. Progress in Theoretical Computer Science. Cambridge, Masssachusetts: Birkhäuser Boston, (1993).

[76] I. Skliarova, A. de Brito. Reconfigurable Hardware SAT Solvers: A Survey of Systems. IEEE Transactions on Computers, Vol. 53, No. 11, pages 1449-1461, (2004)

[77] S.Vadhan. The Complexity of Counting.Bachelor‘s thesis.Mathematics and Computer Science,Harvard College,Cambridge,Massachusetts,April (1995).

[78] SP.Vadhan. The complexity of counting in sparse, regular, and planar graphs. http://theory.lcs.mit.edu/~salil/papers/sparse-regular.ps.gz,(Preprint) pages 55,104,130 (1997).

[79] SP.Vadhan. The complexity of counting in sparse, regular, and planar graphs. http://theory.lcs.mit.edu/~salil/papers/sparse-regular.ps.gz, pages 1-28 (2000).

[80] L.G. Valiant. The complexity of enumeration and reability problems. SIAM Journal of Computing, pages 410-421, (1979).

[81] M. N. Velev. Exploiting Signal Unobservability for Efficient Translation to CNF in Formal Verification of Microprocessors,Proceedings of the Design, Automation and Test in Europe Conference and Exhibition IEEE. 266-271 (2004).

[82] M. Wooldridge,Intelligent Agents: The Key Concepts, Multi- Agent Systems and Applications II, Springer 9th ECCAI-ACAI/EASSS, pages 3-41,(2001).

[83] H. Xu, R.A. Rutenbar, K. Sakallah, sub-SAT: A formulation for related boolean satis.ability with applications in routing. ISPD02, San Diego, CA. (2002).