Lógica Computacional Proyecto 1: "El Problema SAT"

Profesor: Fernando Abigail Galicia Mendoza

Integrantes: Muñoz Barón Luis Miguel
Gallegos Salgado Leslie Paola

22 Febrero 2019

"...more than a body of knowledge. It's a way of thinking. A way of skeptically interrogating the universe with a fine understanding of the human fallability. If we are not able to ask skeptical questions to interrogate those who tell us that something is true, to be skeptical of those in authority then we are up for grabs for the next charlatan political or religious who comes ambling along."
CarlSagan

1 Introducción

La lógica ha sido objeto de estudio desde hace ya bastantes años, teniendo sus primeros intentos de formalización en India, China y Grecia, siendo estos últimos quienes sentaron las bases con la lógica aristotélica y la lógica de predicados con los estóicos. Buscando así construir un mecanismo de inferencia por el cuál se pudiera decidir la veracidad de una afirmación a partir de premisas.

1.1 La lógica matemática

En escencia, la Lógica es el estudio de la argumentación formal, es el seguimiento de una secuencia de afirmaciones que nos permite construir otra a partir de antecedentes. Proporcionando así una herramienta del pensamiento en con la cual se puede modelar nuestro entorno y constituye la base para el razonamiento humano.

Hoy en día la relación entra la Lógica y la computación es tan íntima que resulta difuso dibujar una línea que separe una de la otra. La Lógica no solo es una herramienta de la computación si no una base para la misma, sin élla el

compúto moderno no sería viable o siquiera posible.

1.2 La lógica en las Ciencias de la Computación

Sus aplicaciones se hallan embedidas tanto en la investigación como en la industria, ya que su versatilidad y relación con problemas cotidianos, como su nivel de abstracción resulta idóneo para resolver problematicas y construir tecnologías de los tiempos actuales, aún cuando sus inicios datan de hace más de dos mil años (Aristotélēs; Estagira, 384 a. C.-Calcis, 322 a. C).

Entre las aplicaciones más importantes se encuentran el desarrollo de AI, inferencia de haplotipos en ciencias genómicas y la verificación de modelos y circuitos integrados.
El procedimiento para comprobar la consistencia de un círcuito integrado es como sigue; se toman dos copias de un mismo circuito y se somenten a evaluación asignando valores, si ambos circuitos son idénticos sus salidas deberían en teoría ser la misma, para esto se comparan sus respectivos resultados mediante una interpretación, si el resultado v es el esperado, el circuito es correcto, en otro caso una copia está mal construida.

Otra aplicación es en las bases de datos y los Queries de busqueda, los Queries pueden ser escritos en el lenguaje en el que esté construida la base de datos pero independientemente al lenguaje puede ser expresado como una fórmula lógica con cuantificadores, proposiciones y predicados. Por ejemplo, si se tienen tres tablas de datos digamos Amigos, Novios y Compañeros de clase. Podemos expresar predicados mediante las relaciones que establecemenos entre las tablas, por ejemplo: Desamos encontrar alguna persona que sea amigo de alguien, novio de alguien más y compañero de clase de alguna otra persona podemos especificar un Query de busqueda en términos de la Lógica como sigue:

Amigos:

Molly	Kelly
Luis	Miguel
Leslie	Paola

Novios:

Kelly	Abner
Daniela	Luis
Paola	Fernando

Compañeros de clase:

Paola	Luis
Fernando	Molly
Kelly	Leslie

Supongamos que queremos encontrar a aquella persona que sea Amiga de Molly, novio o novia de Abner y compañero o compañera de clase de Leslie, podemos traducir nuestra busqueda a:

$\exists x(amigos(x, Molly) \wedge novios(x, Abner) \wedge companeros(x, Leslie))$
Ahora, ¿cómo podemos comprobar que existe dicha persona o no? La forma rudimentaria y poco eficiente sería reemplazando x con cualquiera de las personas y evaluando para ver si la proposición es verdadera (Poco eficiente porque en el peor de los casos por el tamaño de las tablas podría caer en $O(n^2)$). Nótese que es equivalente a preguntarnos si existe algún valor para x tal que nuestro Query se interprete como verdadero. Lo que nos lleva a plantearnos el Problema de la Satisfacibilidad Booleana o Boolean Satisfiability Problem que se enuncia como sigue:
Dada una fórmula booleana decimos que es satisfacible si existe una interpretación I() tal que al evaluar dicha fórmula sea verdadera, i.e. si existen determiados valores true/false tal que al reemplazar las variables de nuestra fórmula booleana evalúe a verdadero.

1.3 El problema de la Satisfacibilidad Booleana

El saber cuando una fórmula booleana es satisfacible resulta de gran utilidad en para las ciencias de la computación ya que nos ayuda a probar la consistencia de un circuito, cuando un Query puede ser satisfacido o por ejemplo, cuando los requerimientos de un sistema de estado finito son consistentes, i.e. se puede construir.

Estados:
-El programa no está suspendido o no está en modo multithreading.
-Si el programa no está en modo multithreding entonces está en uso.
-El programa no está está suspendido si y solo si está en uso.
-El programa está suspendido.
Podemos asociar una representación logica al estado del programa
s = suspendido, m = multithreding, u = uso.
$\neg s \vee \neg m$, $\neg m \rightarrow u$, $\neg s \iff \neg s$ s.
veamos el desarrollo de las formulas
$\neg s \vee \neg m$, s entonces $\neg m$
$\neg m \rightarrow u$, entonces u
$u \iff \neg s$, entonces $\neg s$
de aquí surge una contradicción ya que empezamos con el estado S y terminamos concluyendo $\neg s$ por lo tanto el sistema no se puede construir, porque no existe ninguna interpretación tal que I(programa) = 1, porque tenemos algo de

la forma $s \wedge \neg s$ que es una contradicción.
Con los sistemas de resolución SAT el problema anterior es facilmente modelable y facilmente decidible, constituyendo así una herramienta útil y práctica para este tipo de problemas.

En resumen la importancia de la lógica en la Ciencia de la Computación radica en que nos ayuda a entender y trabajar problemas de una manera intuitiva y sencilla como en los dos ejemplos anteriores, gracias a éllo podemos construir nuevas tecnologías o validar aquellas que ya tenemos implementadas.

2 Lógica Proposicional

Una vez que sabemos que podemos modelar con la lógica proposicional resulta conveniente entonces dar la definición formal de la misma, la cual usaremos para modelar problemas lógicos de la vida cotidiana y evaluarlos para saber bajo que condiciones éstos son verdaderos o falsos.

Primero que nada tenemos que definir el lenguaje propocional (PROP), el álfabeto de PROP cuenta con los siguientes símbolos:
Variables proposicionales: generalmente denotadas por las letras "p" y "q" y pueden estar indexadas, tantas como nosotros queramos $p_1 ... p_n$
Constantes Lógicas $\top \bot$.
Operadores $\vee \wedge \iff \rightarrow \neg$.
Denotamos ATOM como las variables p,q y las constantes $\top \bot$.
. Definimos recursivamente a PROP como sigue
1. Si p $\in$ ATOM, entonces p $\in$ PROP.
2. Si $\phi \in$ PROP, entonces $\neg\phi \in$ PROP.
3. Si $\phi, \psi \in$ PROP, entonces $\phi \rightarrow \psi, \phi \vee \psi, \phi \wedge \psi, \phi \iff \psi \in$ PROP.

2.1 Cómo lidiar con la ambigüedad del lenguaje

Ahora bien, una vez definido nuestro lenguaje proposicional tenemos que dar reglas para la interpretación y lectura del mismo, por ejemplo en la expresión $p \rightarrow q \vee r \rightarrow \neg s$ cómo se lee? cuál se tiene que evaluar primero y cuál después? Una solución es utilizar paréntesis, pero puede llegar a ser abrumadora la lectura de las formulas si hay demasiadas presencias de paréntesis:
$(p \rightarrow ((q \vee r) \rightarrow \neg(s)))$

Para evitar esto se estableció la prescendencia de operadores enlistada como sigue:

1) $\neg$ (No, es cierto que, es falso)
2) $\vee, \wedge$ ($\wedge$ se lee como;"y, pero", mientras que $\wedge$ se lee como; "ó")
3) $\rightarrow$ (x $\rightarrow$ y se lee como: "x implica que y", "entonces pasa", o bien "p es condición suficiente para y", "y es condición necesaria para x", "y si q")
4) $\iff$ ($x \iff y$ se lee como: "x si y solo si y", "x es condición necesaria y suficiente para y").

Adicionalmente se convino la asociación de $\rightarrow$ hacia la derecha i.e. $p \rightarrow q \rightarrow s$ $= (p \rightarrow (q \rightarrow s))$.
De igual forma no se trabaja con proposiciones que tengan operadores de la misma precendencia y se considera incorrecto su uso, por ejemplo $p \vee q \wedge s$.
Por ejemplo:

$p \vee q \rightarrow s \iff \neg r \vee t \rightarrow v \rightarrow v$
$= p \vee q \rightarrow s \iff \neg(r) \vee t \rightarrow v \rightarrow v$
$= (p \vee q) \rightarrow s \iff (\neg(r) \vee t) \rightarrow v \rightarrow v$
$= (p \vee q) \rightarrow s \iff (\neg(r) \vee t) \rightarrow (v \rightarrow v)$
$= ((p \vee q) \rightarrow s) \iff ((\neg(r) \vee t) \rightarrow (v \rightarrow v))$

2.2 La abstracción del lenguaje

El lenguaje abstracto es aquel que no tiene un cuerpo, volumen o consistencia material y por lo tanto no se puede representar por medio de un ícono o figura.

Un ejemplo son las figuras geométricas, no existen como tal en el mundo natural pero tienen una representación figurativa simple y universal con la que podemos interpretar expresiones ya sean básicas o complejas.

Una interpretación a diferencia de una representación reconstruye la realidad material atribuyendo un significado determinado, donde podemos determinar su veracidad o falsedad.

Una definición más formal para una Interpretación, basada en el libro "Matemáticas Discretas":

Def. Un estado de las variables proposicionales es una función I que asigna a cada variable proposicional el valor de falso o verdadero.

I: V Prop $\longrightarrow \{0, 1\}$ donde V Prop es el conjunto de variables proposicionales.

Def. Cada estado I determina una interpretación de las formulas -denotada también por I- definida:

I(true) = 1 I(false) = 0
I($\neg P$) = 1 si y sólo si I(P) = 0

$I(P \vee Q) = 0$	si y sólo si	I(P) = 0 = I(Q)
$I(P \wedge Q) = 1$	si y sólo si	I(P) = 1 = I(Q)
$I(P \rightarrow Q) = 0$	si y sólo si	I(P) = 1 e I(Q) = 0
$I(P \leftrightarrow Q) = 1$	si y sólo si	I(P) = I(Q)

Un modelo de una fórmula F es una interpretación, si F es 1 con respecto a esa interpretación.

Ejemplos:
Con las fórmulas:
F= $p \vee q \rightarrow \neg p \wedge q$
I(p) = 1
I(q) = 1
Hace que $I(p \vee q \rightarrow \neg p \wedge q) = 1$, por lo tanto es un modelo para la fórmula.

F = $(p \wedge q \wedge \neg p) \rightarrow (r \vee s \vee t)$
I(p) = 1
I(q) = 1
I(r) = 1
I(s) = 1
I(t) = 1
Hace que $I((p \wedge q \wedge \neg p) \rightarrow (r \vee s \vee t)) = 1$, por lo tanto es un modelo para la fórmula.

F = $(p \wedge (q \wedge r)) \rightarrow (p \rightarrow (q \rightarrow r))$
I(p) = 1
I(q) = 1
I(r) = 1
Hace que $I((p \wedge (q \wedge r)) \rightarrow (p \rightarrow (q \rightarrow r))) = 1$, por lo tanto es un modelo para la fórmula.

2.3 Razonamiento Ecuacional

En lógica computacional nos interesa poder realizar evaluaciones acerca de modelos sobre fórmulas o asignación de valores a las variables de una fórmula, sin embargo, este proceso es caro computacionalmente, por ejemplo. si tenemos una fórmula con 10 variables y queremos obtener todas asignaciones a las 10 variables de nuestra fórmula proposicional tendríamos que hacer 2^{10} asignaciones (2 valores; verdadero o falso por cada una de nuestras 10 variables), esto es 1024 asignaciones, Podemos entonces introducir el concepto de equivalencia.

Decimos que ϕ es equivalente a ψ y lo denotamos como $\phi \equiv \psi$ si $I(\psi) = I(\phi)$
$\forall I$

En cambio, queremos encontrar una forma de evaluar una expresión más corta que nos tome una cantidad menor de pasos, es decir, una expresión equivalente aquí es en donde entra el razonamiento ecuacional.

Teniendo en mente la idea de evaluar una proposición lógica con alguna equivalencia que nos cueste menos esfuerzo y recursos podemos trabajar una expresión con las siguientes reglas y equivalencias de tal forma que nos quede otra más compacta.

Propiedades:

Regla de Leibniz: Si ϕ, ψ, $X \in$ Prop y $p \in$ VarP con $\phi \equiv \psi$
entonces $X[p := \phi] \equiv X[p := \psi]$

Conmutatividad: $\phi \wedge \psi \equiv \psi \wedge \phi$ $\quad$ $\phi \vee \psi \equiv \vee \vee \phi$

Asociatividad: $\phi \vee (\psi \vee X) \equiv (\phi \vee \psi) \vee X$ $\quad$ $\phi \wedge (\psi \wedge X) \equiv (\phi \wedge \psi) \vee X$

Distributividad: $\phi \vee (\psi \wedge X) = (\phi \vee \psi) \wedge (\psi \vee X)$
$\phi \wedge (\psi \vee X) = (\phi \wedge \psi) \vee (\psi \wedge X)$

Leyes de Negación:

. Doble negación: $\neg(\neg\phi) \equiv \phi$
. Demorgan: $\neg(\psi \wedge \phi) \equiv \neg\psi \vee \neg\phi$
. $\neg(\psi \vee \phi) \equiv \neg\psi \wedge \neg\phi$
. Implicación: $\neg(\phi \to \psi) \equiv \psi \wedge \neg\psi$
. Doble Implicación: $\neg(\phi \iff \psi) \equiv \neg\phi \iff \psi \equiv$
. $\phi \iff \neg\psi$

Eliminación de Conectivos:

. $\phi \to \psi \equiv \neg\phi \vee \psi \equiv \neg(\phi \wedge \neg\psi)$
. $\psi \iff \psi \equiv (\phi \to \psi) \wedge (\psi \to \phi) \equiv (\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$
. $\phi \vee \psi \equiv \neg\phi \to \psi$
. $\phi \wedge \psi \equiv \neg(\phi \to \neg\psi)$

Leyes Simplificativas:

Indempotencia: $\phi \wedge \phi \equiv \phi$ $\quad$ $\phi \vee \phi \equiv \phi$
Absorción: $\phi \vee (\phi \wedge \psi) \equiv \psi$ $\quad$ $\phi \wedge (\phi \vee \psi) \equiv \psi$

Neutros: $\phi \vee \bot \equiv \psi \qquad \phi \wedge \top \equiv \psi$
Inversos:
Tercer excluido: $\phi \vee \neg\psi \equiv \phi$
Contradicción: $\phi \wedge \neg\phi \equiv \bot$
Dominancia: $\phi \wedge \bot \equiv \bot \qquad \phi \vee \top \equiv \top$

Con las equivalencias anteriores es fácil ver que podemos reducir el tamaño de una proposición, ya sea sustituyendo partes de la misma por fórmulas más pequeñas o directamente por un valor.
Ejemplo:
$(p \rightarrow \neg q) \vee (p \wedge q) \wedge (\neg s \vee t)$
$= (\neg p \vee \neg q) \vee (p \wedge q) \wedge (\neg s \vee t)$ (Eliminación de la implicación).
$= \neg(p \wedge q) \vee (p \wedge q) \wedge (\neg s \vee t)$ (Demorgan)
$= \top \wedge (\neg s \vee t)$ (Tercer excluido).
$= (\neg s \vee t)$ (Identidad).

Del ejemplo anterior es claro que la fórmula puede reducirse a solo evaluar un pedazo de la misma si utilizamos las equivalencias y las reglas anteriormente enlistadas.

3 Argumentos a fórmulas

En esta sección la meta es aterrizar todos los conceptos de la lógica proposicional para aplicarla en algo un poco menos abstracto, algo que todos utilizamos como la argumentación cuando tratamos de explicar algo o queremos defender una postura.
La lògica proposicional provee una herramienta que nos permite determinar cuando un argumento es correcto o no. Si tenemos un conjunto Γ (que utilizaremos como premisas) y una proposición ϕ (que será nuestro consecuente) decimos que ϕ es consecuencia lógica de Γ y, cuando para toda Interpretación bajo la cual I(Γ) es verdadera I(ϕ) también lo es. Lo denotamos como $\Gamma \vDash \phi$

Los siguientes son dos teoremas que nos ayudaran a verificar la validéz de un argumento.

Principio de Refutación $\Gamma \vDash \phi$ si y solo si $\nvDash \Gamma \bigcup \{\neg\phi\}$
Demostración: Por hipótesis si I(Γ) = 1 entonces I(ϕ) = 1 $\forall I$ Interpretación. entonces I($\neg\phi$) = 0 tenemos así que I(Γ) = 1 y I($\neg\phi$) = 0, entonces I($\Gamma \cup \neg\phi$) = 0 porque no se cumple $\forall\psi \in \Gamma \bigcup \neg\phi$ la interpretación I(ψ) = 1, en particular cuando $\psi = \phi$ I(ψ) = 0 por lo tanto $\nvDash \Gamma \cup \{\neg\phi\}$
. Ahora tomaremos la hipótesis $\nvDash \Gamma \cup \{\neg\phi\}$ p.d que $\Gamma \vDash \phi$ Procedemos por casos Caso 1) I(Γ) = 0, entonces $\Gamma \vDash \phi$ (Insatisfacibilidad implica trivialidad)

Caso 1) I(Γ) = 1, entonces I($\neg\phi$) = 0, así I(ϕ) = 1, entonces se tiene por la definición de satisfacibilidad que $\Gamma \vDash \phi$

Teorema de la deducción Sea Γ un conjunto de proposiciones y $\alpha, \beta \in Prop$
$\Gamma \cup \alpha \vDash \beta$ si y solo si $\Gamma \vDash \alpha \to \beta$
Demostración primero haremos la primera implicación: Tenemos por hipótesis que $\Gamma \cup \alpha \vDash \beta$, así sí I($\Gamma \cup \alpha$) = 1 entonces I($\beta$) = 1 así como I($\Gamma \cup \alpha$) = 1 entonces I($\Gamma$) = 1 y I($\alpha$) = 1, se tiene entonces que I(α) = 1 y I(β) = 1, entonces se cumple que I($\alpha \to \beta$) = 1

Ahora tomamos por hipótesis $\Gamma \vDash \alpha \to \beta$
Sí $\Gamma \vDash \alpha \to \beta$ entonces si si tenemos que I($\Gamma \cup \alpha$) = 1, esto implica que I(Γ) = 1 y I(α) = 1, pero por hipótesis si I(Γ) entonces I($\alpha \to \beta$) = 1, y como ya sabemos que I(α) = 1 entonces I(β) = 1 forzosamente, así $\Gamma \cup \alpha \vDash \beta$.

Con estos dos resultados podemos ahora decir cuando un argumento es correcto o no a continuación mostraremos un ejemplo

El Perro ladra($\neg P$) y come Croquetas(Q)
Si come croquetas(Q), entonces el perro Respira (R)
El perro no ladra ($\neg P$), entonces el perro Respira (R)
Podemos reescribir nuestras proposiciones de la sigueinte forma.

$\neg P \wedge Q$
$Q \to R$
$\neg P$

R

Para poder decidir si nuestro argumento es correcto, entonces hay que aplicar el principio de refutación.
Sea $\Gamma = \{P \wedge Q, Q \to R, \neg P\}$ queremos demostrar que $\Gamma \vDash R$, es decir que $\forall$ Interpretación I tal que I(Γ) = 1 entonces I() = 1, por el principio de refutación, sabemos que $\Gamma \vDash R$ si $\nvdash \Gamma \cup \neg$. Agregamos entonces a las premisas el consecuente negado y tenemos que ver que nuestra $\Gamma' = \{P \wedge Q, Q \to R, \neg P, \neg R\}$ sea insatisfacible, sabemos que I($\neg P$) = 1 entonces I(P) = 0, como I($\neg P \wedge Q$) = 1 entonces I(Q) = 1, así como I($Q \to R$) = 1 entonces I(R) = 1 pero tenemos también que I($\neg R$) = 1 lo cual es una contradicción que nace de suponer que Γ' es satisfacible, por lo tanto se cumple.

4 Forma Normal Conjuntiva

4.1 Equivalencias

Las formas normales nos permiten trabajar con proposiciones más 'manejables' pero que en escencia siguen teniendo el mismo valor proposicional, definimos entonces las equivalencias como: $\phi \equiv \psi$ si $\forall$ Interpretación I se tiene que $I(\phi) = I(\psi)$.
Con esto se pretende buscar un algoritmo, heurística o procedimiento que nos ayude a optimizar el problema de la satisfacibilidad de una proposición. Ya que es mucho mas sencillo trabajar con equivalencias de proposiciones que estén de alguna forma normalizadas o cumplan con alguna propiedad, ya que se posee mas información sobre el contenido de las proposiciones sin perder su valor u contenido a ultranza.
Por ejemplo, la siguiente expresión es equivalente en contenido pero relativamente más sencillo evaluar una expresión cuando está normalizada a alguna forma:

$$\neg(\phi \wedge \psi) \rightarrow \rho \wedge \psi \equiv \psi \wedge (\phi \vee \rho)$$

que es mucho más sencilla de evaluar ya que está en forma normal conjuntiva (FNC).

4.2 Forma Normal Negativa

Una vez que hemos analizado las equivalencias proposicionales resulta natural preguntarnos si hay una forma de expresar fórmulas proposicionales como una familia de otras fórmulas que compartan características entre sí o cumplan con alguna característica, por ejemplo estar libres de presencias de los operadores $\Longleftrightarrow \rightarrow$, o que tengan todos los operadores de negación solo en variables atómicas. La respuesta es sí, de hecho a la familia de proposiciones que cumple con las dos características anteriores se le conoce como la forma normla conjuntiva.

Para poder expresar cualquier fórmula proposicional por medio de cláusulas, que son un conjunto de disyunciones, se utiliza la Forma Normal Conjuntiva.

Definición. Una literal es una fórmula atómica (variable proposicional $p, \bot$ o $\top$) o la negación de una fórmula atómica.

Gramática de literal, cláusula y forma normal conjuntiva

$a ::= \top \mid \bot \mid p$	atómicas
$l ::= a \mid \neg a$	literales
$C ::= l \mid lC$	cláusulas
$F ::= C \mid C \wedge F$	formas normales conjuntivas

4.3 Cómo generar una proposición en Forma Normal Negativa

En este punto tenemos las herramientas necesarias para poder establecer un método que nos transforme cualquier proposición a su forma normal negativa.

Se obtiene una fórmula equivalente donde los símbolos de negación únicamente afectan a las fórmulas atómicas, sin implicaciones ni bicondicionales.

Para que sea válida la equivalencia requiere del cumplimiento de sus leyes de negación:

$Doble\ Negación : \neg\neg\varphi \equiv \varphi$
$De\ Morgan : \neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
$De\ Morgan : \neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
$\neg(\varphi \to \psi) \equiv \varphi \wedge \neg\psi$
$\neg(\varphi \leftrightarrow \psi) \equiv \neg\varphi \leftrightarrow \psi \equiv \varphi \leftrightarrow \neg\psi$

Algoritmo para tranformar una fórmula a su forma normal negativa:
1. Eliminar implicaciones y bicondicionales respectivamente con las siguientes equivalencias:

$$\varphi \to \psi \equiv \neg\varphi \vee \psi$$
$$\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$$

2. Usar *De Morgan* y *Doble Negación* para obtener una fórmula simplificada con negaciones afectando sólo a las literales.

De esta forma se puede generar una fórmula con características específicas, ya sea eliminando las dobles negaciones, implicaciones y/o bicondicionales.

4.4 Como generar una proposición en Forma Normal Conjuntiva

La Forma Normal Negativa no es la única forma que nos será de utilidad para determinar cuando una fórmula es satisfacible o no, de hecho vamos a utilizarla como herramienta para construir la siguiente forma normal que nos interesa para determinar la satisfacibilidad de una fórmula.
Esta forma es la Forma Normal Conjuntiva que se define como sigue:

Vamos a definir a la Forma Normal Conjuntiva con la siguiente gramática:

1. a ::= $\top|\bot|p$ atómicas
2. l ::= $\alpha|\neg\alpha|$ literales

3. C ::= $l|l \vee C|$ cláusulas

4. F ::= $C|C \wedge F|$

Proposición; Para toda $\phi \in$ Prop existe una $\psi \in$ Prop tal que $\phi \equiv \psi$ y fnc(ψ) y se puede obtener algorítmicamente.
Demostración: Basta con transformar en orden:

1. Obtener la Forma Normal Negativa de ϕ

2. Aplicar las leyes distributivas de las conjunción y disyunción
   $(\phi \vee \psi) \wedge (\phi \vee \rho) = \phi \vee (\psi \wedge \rho)$
   $(\phi \wedge \psi) \vee (\phi \wedge \rho) = \phi \wedge (\psi \vee \rho)$
   Del caso general se tiene que $(a_1, \vee ..., \vee a_n) \wedge (a_1, \vee ..., \vee b_n) = (a_1 \wedge b_1) \vee ... \vee (a_1 \wedge b_n) \vee (a_2 \wedge b_1) \vee ... \vee (a_2 \wedge b_n) \vee ... \vee (a_n \wedge b_1) \vee ... \vee (a_n \wedge b_n)$

3. Opcionalmente se puede simplificar aplicando las equivalencias de la sección 2.3

Algoritmo para Obtener la FNC de una proposición:

Nuestro algoritmo recibe una fórmula ϕ y regresa como salida uan fórmula ψ tal que $\phi \equiv \psi$ con ψ en forma normal conjuntiva
Caso 1) $\phi =$ p ya está en forma normal negativa, entonces $\psi =$ p.
Caso 2) $\phi = \phi_1 \wedge \phi_2$ entonces se $\psi = fnc(\phi_1) \wedge fnc(\phi_2)$ si $\phi = \phi_1 \vee \phi_2$ entonces $\psi =$ = distr($\phi_1 \wedge \phi_2$)

Donde distr :: Prop -> Prop -> Prop y se toman las siguientes concideraciones.
1) Asumimos que la las dos Proposiciones que recibe $\phi_1\phi_2$ están en forma normal conjuntiva.
2) si ambas $\phi_1 y \phi_2$ son literales entonces regresamos $\phi_1 \vee \phi_1$
3) Si $\phi_1 = \phi_{11} \wedge \phi_{12}$ entonces aplicamos la distribución del conectivo $\wedge =$ $(\phi_{11} \wedge \phi_{12}) \wedge \phi_2 \equiv (\phi_{11} \vee \phi_2) \wedge (\phi_{12} \vee \phi_2)$

Ejemplo: Sea $\phi = ((p \wedge r) \rightarrow p) \wedge (q \rightarrow (s \vee q))$
vamos a pasarla a FNC
1) Pasar a Forma Normal Negativa (eliminar implicación y distribuir las negaciones hasta las atómicas).
$= (\neg(p \wedge r) \vee p) \wedge (\neg q \vee (s \vee q))$
$= ((\neg p \vee \neg r) \vee p) \wedge (\neg q \vee s \vee q)$
$= (\neg p \vee \neg r \vee q) \wedge (\neg q \vee p \vee s)$
ésta es la Forma Normal Conjuntiva

Ahora para hallar si esta fórmula es satisfacible, resulta mucho más sencillo

que antes de aplicar la transformación, porque como tenemos que la Forma normal Conjuntiva es una disyunción de conjunciones $(C_1, \wedge C_2 \wedge ...C_n)$ entonces con que una conjunción sea falsa resulta insatisfacible el resto de la proposición. Como nuestra propsición en FNC es una conjunción de cláusulas entonces ambas cláusulas dentro de la conjunción deben evaluar a verdadero, ésto lo logramos con atómicas contrarias, esto es porque $\top = \neg\phi \vee \phi$
$(\neg p \vee \neg r \vee p) \wedge (\neg q \vee s \vee q)$
$=(\neg p \vee p \neg r) \wedge (\neg q \vee q \vee s)$
$=((\neg p \vee p)\neg r) \wedge ((\neg q \vee q) \vee s)$
$=(\top \neg r) \wedge (\top \vee s)$
$=\top \wedge \top$
$=\top$
Por lo tanto es satisfacible
En cambio si evaluamos la interpretación 'a mano' de la proposición que no está en FNC sería así:
I($((p \wedge r) \rightarrow p) \wedge (q \rightarrow (s \vee q))) = 1$ si y solo sí I($(p \wedge r) \rightarrow p) = 1$ y I$(q \rightarrow (s \vee q))$ $= 1$ Sí I$((p \wedge r) \rightarrow p) = 1$ entonces asumimos que I(p) = 1, y para que la implicación sea verdadera tiene que pasar que I$(p \wedge r) = 1$ si solo sí I$(p) = 1$, I$(r) =$ 1
Ahora lo hacemos para I$(q \rightarrow (s \vee q))$
I$(q \rightarrow (s \vee q)) = 1$
Asumimos que I$((s \vee q)) = 1$, entonces I$(s) = 1$ e I$(q) = 1$
entonces I$(q \rightarrow (s \vee q)) = 1$
En particular nuestra fòrmula va a ser satisfacible con las Interpretaciones I(p) $= 1$, I$(r) = 1$, I$(q) = 1$, I$(s) = 1$, por lo tanto es satisfacible.

La FNC brinda una herramienta para encontrar cuando una proposición es satisfacible de forma más sencilla aunque esto implica aumentar el número de conectivos. La ventaja es que podemos aplicar el método de Resoulución proposicional que se discute en la siguiente sección.
Del ejemplo anterior es fácil ver que una fórmula en FNC nos permite detener la evaluación cuando encontramos variables contrarias.

4.5 SAT y la Forma Normla Conjuntiva

El hecho de tener una proposición en su Forma Normal Conjuntiva nos permite establecer 'atajos' para determinar cuando será satisfacible o no (Como el hecho de tener las variables contrarias del ejemplo anterior en una cláusula). Más aún, nos permite definir un algoritmo que será útil para determinar su satisfacibilidad. En la siguiente sección definiremos un método que nos permite deducir cuando una formúla es satisfacible utilizando las propiedades de la FNC

4.6 Resolución Proposicional

Para determinar la correctud de un argumento lógico también contamos con la *regla de resolución binaria* de Robinson, donde lleva por nombre *binaria* por

sus dos premisas, donde se toman dos cláusulas tales que existe una literal ℓ que figura en una de ellas, mientras que la literal contraria ℓ^C figura en la otra.
Es importante aclarar que esta regla opera únicamente sobre un par de literales complementarias a la vez y no con más.

Definición: Donde $C1$ y $C2$ son cláusulas y ℓ una literal con su respectivo complemento ℓ^C.

$$\frac{C1 \vee \ell \qquad \ell^C \vee C2}{C1 \vee C2}\ \text{RES}$$

Se le llama *refutación del conjunto de cláusulas* al proceso de repetición de la regla de resolución binaria hasta obtener la cláusula vacía, denotada como $\square$.

Para decidir si $\Gamma \models \varphi$ basta demostrar que $\Gamma \cup \{\neg\varphi\}$ es insatisfacible.

Gracias a la *regla de resolución binaria*, para generar todas las derivaciones con resolución a partir de un conjunto dado ($\mathbb{S}$) definimos el algoritmo de saturación:

Definición : Si $\mathbb{S}$ es cualquier conjunto de cláusulas entonces la resolución de $\mathbb{S}$, denotada $R(\mathbb{S})$ es el conjunto que consiste de $\mathbb{S}$ junto con todoslos resolventes de cláusulas de $\mathbb{S}$.

$$Res_0(\mathbb{S}) = \mathbb{S}$$
$$Res_{n+1}(\mathbb{S}) = R(Res_n(\mathbb{S}))$$

donde:
Sea $\mathbb{S}$ un conjunto finito de cláusulas. Entonces $\mathbb{S}$ es no satisfacible si y sólo si $\square \in Res_n(\mathbb{S})$para alguna $n \in \mathbb{N}$.

Ejemplo de algoritmo de saturación para una fórmula FNC:

$(p \vee q \vee \neg r) \wedge (\neg q \wedge r) \wedge \neg p$

$Res_0 = \{\ p \vee q \vee \neg r, \neg q, r, \neg p\}$
$Res_1 = Res0 \cup \{p \vee \neg r, p \vee q, q \vee \neg r\}$
$Res_2 = Res1 \cup \{p, \neg r, p, q, q, \neg r, q, ...\}$
$Res_3 = Res2 \cup \{\square, \square, \square, ...\}$

En otro caso el algoritmo de saturación no puede decidir la satisfacibilidad de una fórmula cualquiera si en el conjunto de cláusulas en FNC no existe alguna ℓ y ℓ^C, ya que nunca concluiremos $\square$.
Por ejemplo:
$\mathbb{S} = \{p \to q, r \wedge q\} \equiv \{\neg p \vee q, r \wedge q\}$
$Res_0 = \{\neg p \vee q, r, q\}$

$Res_1 = Res_0 \cup \{\neg p \vee q, \neg p \vee q\}$
$Res_2 = Res_1 \cup \{\neg p \vee q, \neg p \vee q, \neg p \vee q, \neg p \vee q\}$
...
...

Si continúa el algoritmo infinitamente nunca llegará a una cláusula vacía por lo que no podrá verificar la satisfacibilidad de la fórmula dada.

5 Tableux

Ahora que tenemos un algoritmo para deducir cuando una fórmula es satisfacible queremos encontrar un método más general que nos permita decidir si una fórmula es tautología, contradicción, si un conjunto de fórmulas proposicionales Γ es satisfacible, si una fórmula proposicional ψ es conseciencia lógica de el conjunto de proposiciones Γ o simplemente si una fórmula es satisfacible. Es por éllo que introducimos ahora el metodo de derivación por Tabluex.

La ídea principal de un tableux es generar un árbol cuyos nodos sean proposiciones y sus ramas sean las fórmulas conectadas mediante la variable principal de esta proposición.
queremos repetir el procedimiento hasta tener sòlo variables atómicas en las hojas de nuestro árbol y así generar una interpretación para la fórmila original.

Como lo que que queremos es generar una interpretación para una fórmula dada nos interesa trabajar con las variables atómicas (Entre más simple sea la fórmula que recibimos mejor por que entonces tenemos que hacer menos 'transformaciones') por lo que es conveniente que nuestro Tableux reciba una fórmula en su Forma Normal Conjuntiva (Anteriormente ya se definieron las equivalencias para la implicación y doble condicional). Si la fórmula recibida no está en FNC podemos ir transformando nuestras porposiciones conforme desarrollamos el Tableux con las siguientes reglas:

5.1 Representación de un Tableux

Como ya vimos en la sección de 4.2) - 4.4) toda fórmula proposicional tiene una equivalente en Forma Normal Conjuntiva, por razones de simplicidad nuestro Tableux solo definirá las ramas para dos conectivos a saber $\wedge$ $\vee$ Ejemplo para el caso de una disyunción tendríamos el siguiente árbol

(1) $(P \vee Q)$

(2) P Q

Para el caso de la conjunción la representación es vertical

(1) $(P \wedge Q)$
(2) P
(3) Q

Para simplificar las fórmulas que no estén en Forma Normal Conjuntiva podemos aplicar las siguientes reglas:

$\alpha - reglas$
si tenemos $A \wedge B$ se deduce A y B
si tenemos $\neg(A \vee B)$ se deduce $\neg$A y $\neg$B
si tenemos $\neg(A \rightarrow B)$ se deduce A y $\neg$ B

$\beta - reglas$
si tenemos $A \vee B$ se deduce A y en una rama bifurcada B
si tenemos $\neg(A \wedge B)$ se deduce $\neg$A y en una rama bifurcasa$\neg$B
si tenemos $(A \rightarrow B)$ se deduce $\neg$A y en una rama bifurcada B

$\sigma - reglas$
si tenemos $|neg\neg A$ se deduce A
si tenemos $\neg\bot$ se deduce $\top$
si tenemos $\neg\top$ se deduce $\bot$

Una vez teniendo clara la representación y las reglas que definen a un Tabluex podemos explicar el método de como hallara una interpretación por Tableux. El método radica en encontrar ramas cerradas (aunque pueden no serlo). Una rama cerrada es una rama del árbol de derivación por tableux que contiene a una variable y su complementaria. De manera informal podemos pensar las ramas como una conjunción de las variables atómicas $(p_1 \wedge ... \wedge p_n)$ si encontramos una p_n y $\neg p_n = \bot$ entonces toda la conjunción será $\bot$. Ejemplo de rama cerrada

$P \wedge (Q \vee \neg P)$

(1) P
(2) $(\neg P \vee Q)$
(3) $\neg P$ $\quad$ Q
$\otimes$

En este ejemplo la rama izquierda se cierra porque del camino de ésa hoja a la raíz nos encontramos con P y $\neg P$.

Ahora como interpretamos ésto? las ramas negadas quiere decir que no puedes dar una interpretación tal que esa interpretación con las variables de esa rama evaluen a verdadero tu fórmula original.

5.2 Consideraciones

Al trabajar con Tableux queremos trabajar lo menos posible, es decir desarrollando la menor cantidad de ramas, por ende en cuanto una rama en el tableux se cierre ya no es necesario seguir desarrollando el resto de la rama (hemos encontrado una interpretación que no hace verdadera a la proposición).

1) Descomponer las fórmulas que no abran ramas, es decir aplicar primero las $\alpha - reglas$ segidas de las $\sigma - reglas$ y hasta el final las $\beta - reglas$.

2) Dar prioridad a las fórmulas tal que al expandirlas en las ramas del Tableux cierren ramas (si encontramos dos fórmulas tal que una contiene a la complementaria de una variable en la otra desarrollar primero esas dos).

3) Si no se pueden aplicar ninguna de las reglas anteriores comenzar a desarrollar la poposición más compleja.

5.3 La interpretación de Tableux

Para hallar un modelo solo basta con encontrar una rama abierta, los valores de la interpreteación para cada variable en la rama serían 1 y aquellos que no figuran en esa rama podrían ser 0 o 1, no importa.

Si el Tableux tiene ramas abiertas y cerradas decimos que es una proposición contingente (Puede ser verdadera o falsa dependiendo de los valores que tome la interpretación de sus variables.

Si todas las ramas de tableux son abiernas es una fórmula contingente.

Si todas las ramas del tableux son cerradas entonces es una contradicción.

Si todas las todas las ramas de $\neg\Gamma$ son cerradas entonces Γ es una tautología.

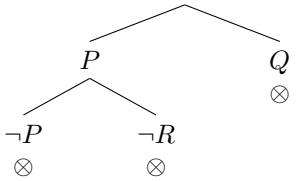
Si Γ es un conjunto de fórmulas hacer el Tableux para $p_1 \wedge ... \wedge p_n$ con $p_i \in \Gamma$, si todas las ramas se cierran entonces es insatisfacible, en otro caso existe una

rama abierta que difne un modelo para Γ

$\Gamma \vDash A$, cosntruir el tableux para $\Gamma \cup \neg A$ si el Tableux cierra todas sus ramas entonces A es consecuencia lógica de Γ.

5.4 Ejemplos de Tableux

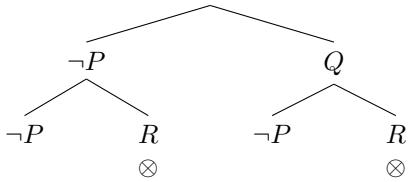
(1) $(P \vee Q) \wedge (\neg P \vee \neg R) \wedge (\neg Q) \wedge (R \wedge \neg S)$
(2) $P \vee Q$
(3) $\neg P \vee \neg R$
(4) $\neg Q$
(5) $R \wedge \neg S$
(6) R
(7) $\neg S$



(9)

Los argumentos son correctos.

(1) $(\neg P \vee Q) \wedge (\neg P \vee R) \wedge (Q \wedge \neg R)$
(2) $(\neg P \vee Q) \wedge (\neg P \vee R)$
(3) $Q \wedge \neg R$
(4) Q
(5) $\neg R$
(6) $\neg P \vee Q$
(7) $\neg P \vee R$



(9)

Los argumentos no son correctos.

References

[1] Miranda Perea F.E y Viso Gurovich E., *Matemáticas Discretas*, Segunda edición, México, UNAM Facultad de Ciencias, 2016.

[2] Miranda Perea F.E, et al *Introducción y Sintaxis de la Lógica de Proposiciones Lógica Computacional*, México, UNAM Facultad de Ciencias,

2017.

[3] Marques-Silva, J. *Practical applications of boolean satisfiability*, 2008 9th International Workshop on Discrete Event Systems (2008), 74–80.

[4] Huth, M., and Ryan, M *Logic in Computer Science: Modelling and Reasoning About Systems.*, Cambridge University Press, New York, NY, USA, 2004.