

# TAREA 1

## PREGUNTA 1: LÓGICA PROPOSICIONAL MATEMÁTICAS DISCRETAS

Integrantes:

- Franco Cattani
- Nicolás del Valle
- Jorge Espinosa

1. Suponga que le entregan un algoritmo de *caja negra* de resolución SAT, es decir, un dispositivo que toma una fórmula de lógica proposicional  $\phi$  y devuelve si  $\phi$  es o no satisfacible. Usted no sabe nada sobre el funcionamiento de este algoritmo. Vamos a denotar este algoritmo  $A$ , por lo que  $A(\phi)$  es verdadero si  $\phi$  es satisfacible.

Esta pregunta plantea qué más se puede hacer con un algoritmo de resolución SAT.

- (a) Cree un algoritmo que utilice  $A$  como subrutina para determinar si  $\phi$  es una tautología. Demuestre que su algoritmo es correcto. No se limite a enumerar todas las posibles asignaciones y comprobar cada una individualmente.

**Respuesta:** Para este algoritmo utilizaremos el siguiente *lemma*

*Lemma*

La proposición  $\phi$  es una tautología si y solo si  $\neg\phi$  es no satisfacible.

En base a esto el algoritmo se le pasa como parametro  $\phi$  para luego utilizar como subrutina  $A$  con el parámetro  $\neg\phi$ , es decir,  $A(\neg\phi)$ , si  $A(\neg\phi) = F$  entonces  $\phi$  es tautología, de lo contrario no es tautología.

- (b) Suponga que tiene dos fórmulas proposicionales  $\phi$  y  $\psi$ . Te interesa determinar si  $\phi \equiv \psi$ , es decir, si  $\phi$  y  $\psi$  tienen siempre los mismos valores de verdad. Crea un algoritmo que utilice  $A$  como subrutina para responder esta pregunta, y demuestra que tu algoritmo es correcto.

**Respuesta:**

Al algoritmo se le pasarán dos parámetros,  $\phi$  y  $\psi$ , luego se guardará el resultado de  $A(\neg(\phi \equiv \psi))$ , este valor de verdad lo llamaremos  $\zeta$ , si  $\zeta = Falso$  entonces  $\phi$  y  $\psi$  siempre tienen el mismo valor de verdad, si por el contrario  $\zeta = Verdadero$  entonces  $\phi$  y  $\psi$  no siempre tienen el mismo valor de verdad. Con esto el algoritmo cumple su función el cual es comprobar la equivalencia entre  $\phi$  y  $\psi$ .

Para demostrar esto nos basaremos en el *lemma* mencionado anteriormente, sabemos que  $\phi \equiv \psi$  es una fórmula proposicional, entonces tiene un valor de verdad.

Para poder determinar si se cumple esta equivalencia  $\phi \equiv \psi$  debe ser una tautología, recordemos que la definición de tautología es;

#### Tautología

Una fórmula  $\alpha \in \xi(P)$  es una **tautología** si y solo si es **verdadera** bajo cualquier valuación, es decir, si y solo si  $\widehat{\sigma}(\alpha) = 1$  para toda valuación  $\sigma : P \rightarrow \{0, 1\}$

La definición de tautología para este caso nos dice que para toda valuación  $P$ ,  $\phi \equiv \psi$  siempre se cumplirá, por ende,  $\zeta = \text{Falso}$  es lo mismo que decir que  $\neg(\phi \equiv \psi)$  es insatisfacible si y solo si  $\phi \equiv \psi$  es una tautología, con lo que el algoritmo cumple con lo pedido.

- (c) Suponga que tiene una fórmula proposicional  $\phi$  con  $n$  variables que sabe que es satisfacible. Cree un algoritmo que utilice  $A$  como subrutina para obtener una asignación satisfactoria para  $\phi$  utilizando como máximo  $n$  llamadas a  $A$ . Demuestre que su respuesta es correcta. El algoritmo tomará como variables  $\phi, n, A$ , supongamos que las  $n$  variables son  $P : \{P_1, P_2, \dots, P_n\}$ , sabemos que son  $2^n$  posibles asignaciones y  $n$  llamadas al algoritmo  $A$  por ende intentar a la fuerza bruta no es factible, por otro lado
2. En esta pregunta usted construirá fórmulas de la lógica proposicional que definen como sumar números binarios.

Considere el conjunto de variables proposicionales

$P = \{a_0, a_1, \dots, a_{n-1}, b_0, b_1, \dots, b_{n-1}\}$ . Podemos suponer que cada valuación  $\sigma : P \rightarrow \{0, 1\}$  define un par de números binarios  $X_1^\sigma$  y  $X_2^\sigma$  dados por la evaluación de  $\sigma$  sobre las secuencias de variables  $a_{n-1} \dots a_1 a_0$  y  $b_{n-1} \dots b_1 b_0$ .

Por ejemplo, si  $n = 3$  y  $\sigma$  es tal que,  $\sigma(a_2) = \sigma(a_1) = \sigma(b_1) = 0$  y  $\sigma(a_0) = \sigma(b_2) = \sigma(b_0) = 1$  entonces  $X_1^\sigma = 001$  y  $X_2^\sigma = 101$ .

Construya fórmulas (solamente con los conectivos  $\{\vee, \neg\}$ )  $\phi_0, \phi_1, \dots, \phi_{n-1}, \phi_n$  en  $\mathcal{L}(P)$  tales que para toda valuación  $\sigma$  se cumpla que  $\sigma(\phi_i) = 1$ , si y solo si, el bit en la posición  $i$  de  $X_1^\sigma + X_2^\sigma$  es 1.

*Sus fórmulas las deben dejar en función de  $\{\neg, \vee\}$ , en caso de que usen algún otro conectivo lógico ( $\{\wedge, \Rightarrow, \oplus, \text{etc}\}$ ), **DEBEN** definirlo en base a  $\{\neg, \vee\}$ , por más trivial que sea (esto es enunciado solamente).*

Para poder definir la suma de números binarios debemos cubrir 2 situaciones las cuales son:

- (a) Suma de bits para cada  $i$  –ésima posición, es decir,  $a_i + b_i$ .
- (b) Cuando  $a_i + b_i + c_{i-1} > 1$  con  $c_{i-1}$  que representa al "bit sobrante" de la suma de  $a_{i-1} + b_{i-1} + c_{i-2}$ .

Notemos que en (b) tenemos un problema para  $i = 0$ , por ende, para  $i = 0$  tomamos  $c_{i_0} = 0$ . Comenzamos definiendo  $c_{i-1}$ : Claramente  $c_{i-1} : \{0, 1\}$  por lo tanto para que este sea 1, suponiendo que  $c_{i-2} = 1$ ,  $a_i = 1$  ó  $b_i = 1$ , por otro lado si  $c_{i-2} = 0$  obliga a  $a_i = 1$  y  $b_i = 1$ , por último para que  $c_{i-1} = 0$  obliga tanto a  $a_i = 0$  y  $b_i = 0$ , por tanto la fórmula proposicional que define lo enunciado es:

$$\mathcal{C}(a_i, b_i, c_{i-1}) = (a_i \wedge b_i) \vee (a_i \wedge c_{i-1}) \vee (b_i \wedge c_{i-1})$$

Para dejar esta fórmula en los conectivos  $\{\neg, \vee\}$  basta con aplicar ley de morgan.

$$\mathcal{C}(a_i, b_i, c_{i-1}) = \neg(\neg a_i \vee \neg b_i) \vee \neg(\neg a_i \vee \neg c_{i-1}) \vee \neg(\neg b_i \vee \neg c_{i-1})$$

Para la suma de bits basta con el operador  $\oplus$  (*XOR*) el cual debemos expresar en los conectivos  $\{\neg, \vee\}$ .

$$\begin{aligned} a_i \oplus b_i \oplus c_{i-1} &\equiv (a_i \oplus b_i) \oplus c_{i-1} \equiv a_i \oplus (b_i \oplus c_{i-1}) \\ &\equiv (\{(a_i \wedge \neg b_i) \vee (\neg a_i \wedge b_i)\} \wedge \neg c_{i-1}) \vee (\{\neg(a_i \vee \neg b_i) \vee \neg(\neg a_i \wedge b_i)\} \wedge c_{i-1}) \end{aligned}$$

Nuevamente para dejar esta fórmula solo con los conectivos indicado más arriba, basta con aplicar ley de morgan.

$$a_i \oplus b_i \oplus c_{i-1} \equiv \neg(\neg\{\neg(\neg a_i \vee b_i) \vee \neg(a_i \vee \neg b_i)\} \vee c_{i-1}) \vee \neg(\{\neg(\neg a_i \vee b_i) \vee \neg(a_i \vee \neg b_i)\} \vee \neg c_{i-1})$$

Con esto podemos sumar números binarios donde tendremos dos fórmulas lógicas las que nos definen la operación suma.