CC3101-1, Matematicas Discretas para la Computacion
Profesor: Alejandro Hevia
Auxiliares: Antonia Labarca & Pablo Torres
Ayudantes: Mauricio Araneda & Francisco Sanhueza & Diego Vargas

Tarea 1

José Pacheco

P1] La función *SetLiteral* crea una lista vacía en la que se van agregando los elementos que nos interesan. Para agregar los elementos se hace mediante dos **for**, uno para las clausulas y otro para las variables proposicionales de cada clausula. Cada vez que se cambia de clausula se crea una lista vacía que después sera agregada (si es que no es vacía) a la lista vacía del inicio. Dentro de los **for** se analiza cada variable, si la variable es igual al literal que se entrego a *SetLiteral*, la lista de clausulas que se van a agregar se cambia a vacía y se sale del **for** (**break**) y si no se agrega a la lista, si la variable es la negación del literal, no se agrega a la lista y se continua con la siguiente variable proposicional (**continue**), una vez que analiza todas las variables de una clausula se agrega la nueva lista a la lista del comienzo, y cuando termina de analizar toda la lista se retorna la nueva lista.

Para la funcion *IsSatisfiable* se comienzan analizando los casos bases, si la lista es vacía retorna **True**, en cambio si dentro de la formula hay dos o mas clausulas de largo uno en las cuales en una esta la negación de la variable proposicional de otra retorna **False**, si ninguno de estos es el caso, toma la primera variable proposicional de la formula (a no ser que haya una clausula de largo uno) y hace una recursion donde la nueva formula es un *SetLiteral* con la formula y el literal seleccionado.

Por ultimo para la función *BuildModel* se prueba si la formula es satisfacible, si no lo es retorna **(False,{})** inmediatamente. Si es satisfacible, se crea un diccionario vació y se crea un **while** para que se ejecute mientras el largo de la formula es distinto de cero, se toma la primera variable proposicional que aparece en la formula y luego probamos si es satisfacible *SetLiteral(formula,primeravariable)*, si lo es, añadimos la variable al diccionario, si es negativa, se agrega positiva con un valor de **False**, si no se agrega la variable con un valor de **True** y despues formula toma el valor del *SetLiteral* hecho anteriormente. Si no es satisfacible, se agrega al diccionario la negacion de la variable y se repite el proceso anterior. Cuando llega el momento en que la funcion tiene largo 0, sale del **while** y retorna **(True,diccionario)**.

P2] Una variable puede tomar el valor de Verdadero o Falso, por lo que el total de combinaciones posibles para **n** variables es de 2^n y además para combinación el resultado también puede ser Verdadero o Falso, por lo que hay 2^{2^n} posibles resultados de salida, lo que implica que para cada salida tiene que existir un operador que de como resultado esa salida, por lo que la cantidad de operadores n-arios que existen para **n** es de 2^{2^n}.

P3] **a)** Primero probaremos la implicancia hacia la derecha:
$\Rightarrow$) Se demostrara por contradicción, es decir $\{\phi_1, ..., \phi_n\} \models \alpha$ si $\phi_1 \wedge ... \wedge \phi_n \rightarrow \alpha$ no es tautología.
Si $\phi_1 \wedge ... \wedge \phi_n \rightarrow \alpha$ no es tautología, significa que existe una interpretación tal que $\phi_1 \wedge ... \wedge \phi_n$ es Verdad y α es Falso, si se tiene eso, α no puede ser consecuencia lógica de $\phi_1 \wedge ... \wedge \phi_n$ por la definición de consecuencia lógica, por lo que se llega a una contradicción lo que implica que $\phi_1 \wedge ... \wedge \phi_n \rightarrow \alpha$ si es tautología.
$\Leftarrow$) Si $\phi_1 \wedge ... \wedge \phi_n \rightarrow \alpha$ es tautología, significa que existen 3 posibles combinaciones para los valores de verdad de $\phi_1 \wedge ... \wedge \phi_n$ y α los cuales son: {V,V},{F,V} y {F,F}, ninguna de estas combinaciones van en contra de la definición de consecuencia lógica, por lo que $\{\phi_1, ..., \phi_n\} \models \alpha$-

b) Supongamos que existen dos conjuntos de formulas de lógica proposicional A y α tal que $A \models \alpha$ y además que existe una interpretación tal que $A = 1$ y $\alpha = 1$. Luego por monotonía se tiene que $A \cup \{\emptyset\} \models \alpha$, pero por propiedad del conjunto vacío $A \cup \{\emptyset\} = A$ por lo que para interpretación que logra $A = 1$ también debe cumplir que $A \cup \{\emptyset\} = 1$ lo que es igual a que $A = 1 \wedge \{\emptyset\} = 1$ por lo que el conjunto vacío es satisfacible.

c) Sea $\mathcal{L}(P)$ el conjunto de todas las formulas de lógica proposicional sobre P significa que si existe ϕ en $\mathcal{L}(P)$ también debe existir $\neg\phi$ (o si no, no estarían todas las formulas proposicionales sobre P) por lo que el conjunto $\mathcal{L}(P)$ contendría $\phi\wedge\neg\phi$ por lo que seria insatisfacible ya que esa conjunción siempre sera falsa para cualquier interpretación.

P4] **a)** Para que el horario no tenga clases a las 8:30, se debe tener que para $j = 1$, y para todos los i, D_{i1}, A_{i1} y G_{i1} deben deben ser 0, lo que se traduce en:

$$\bigwedge_{i=1}^{5} \neg(D_{i1} \vee A_{i1} \vee G_{i1}) \tag{1}$$

La formula anterior tiene valor de 1 solo si D_{i1}, A_{i1} y G_{i1} son falsos para cualquier i, ya que si uno es verdadero la formula tiene un valor de 0 lo que significa que si tiene clases entre las 8:30 y 10:00.

b) Para que el horario no tenga choques, para el mismo par j,i solo un curso puede tener clases, lo que se puede expresar con la siguiente formula:

$$\bigwedge_{i=1}^{5} \bigwedge_{j=1}^{6} \neg(D_{ij} \wedge A_{ij}) \wedge \neg(D_{ij} \wedge G_{ij}) \wedge \neg(G_{ij} \wedge A_{ij}) \tag{2}$$

Esto ya que si dos o mas cursos tienen clases en el mismo horario el resultado de la formula es 0, lo que indica que si tiene choques.

c) Para que el horario no tenga ventanas, se debe tener que si en una hora hay clases y en la siguiente no, no deben haber mas clases en todo lo que sigue del dia, lo que se puede hacer con una implicancia. Para simplificar la notación definiremos $\phi_{ij} = A_{ij} \vee D_{ij} \vee G_{ij}$ y la expresión queda:

$$\bigwedge_{i=1}^{5} \bigwedge_{j=1}^{5} \left((\phi_{ij} \wedge \neg\phi_{i,j+1}) \Rightarrow \bigwedge_{k=j+1}^{6} (\neg\phi_{i,k}) \right) \tag{3}$$