LMD. Práctica 3.
Álgebras de Boole.

1. Ejemplos de álgebras de Boole finitas

Dado un entero positivo n, denotamos por $D(n)$ el conjunto de los divisores positivos de n. En Maxima podemos obtener fácilmente el conjunto $D(n)$ mediante el comando `divisors`. Veamos algunos ejemplos.

```
(%ixx) divisors(8);
```

```
(%ixx) divisors(2014);
```

```
(%ixx) divisors(1);
```

Sabemos que $D(n)$ es un álgebra de Boole si y sólo si n factoriza como producto de (uno ó más) primos distintos, siendo las operaciones $a + b = a \vee b = mcm(a, b)$, $a \cdot b = a \wedge b = mcd(a, b)$ y $\overline{a} = \frac{n}{a}$.

```
(%ixx) factor(12);
```

Por tanto $D(12)$ no es un álgebra de Boole.

```
(%ixx) factor(2014);
```

En vista de la salia obtenida, concluímos que $D(2014)$ es un álgebra de Boole.

```
(%ixx) factor(2017);
```

Vemos pues que 2017 es un primo, y así $D(2017)$ es un álgebra de Boole.

A continuación escribimos una función `esAB(n)`, que se supone se aplica a un entero positivo n, y devuelve `true` si $D(n)$ es un álgebra de Boole, y devuelve `false` en caso contrario.

```
(%ixx) esAB(n):=block([lisf,i,long], lisf:ifactors(n),long:length(lisf),
        if is(apply("*",makelist(lisf[i][2],i,1,long))=1) then true else false)$
```

Probamos la función que acabamos de escribir con los ejemplos anteriores.

```
(%ixx) esAB(12);
```

```
(%ixx) esAB(2014);
```

```
(%ixx) esAB(2017);
```

A continuación calculamos todos los enteros $2 \leq n \leq 50$ tales que $D(n)$ es un álgebra de Boole. Para ello primero creamos el conjunto X formado por todos los enteros entre 2 y 50. A continuación obtenemos el subconjunto Y de X formado por aquellos elementos que hacen que la función `esAB` valga `true`.

```
(%ixx) X:setify(makelist(i,i,2,50))$
```

```
(%ixx) Y:subset(X,esAB);
```

Ahora factorizamos los elementos del conjunto `Y` calculado, y constatamos que cada uno de ellos o bien es primo o es un producto de primos distintos.

```
(%ixx) factor(listify(Y));
```

La técnica de búsqueda empleada en este último ejemplo será utilizada con bastante frecuencia en las prácticas de la asignatura.

Ejercicio 1. Obtenga todos los enteros $100 \leq n \leq 400$ tales que $D(n)$ es un álgebra de Boole.

Ejercicio 2. Obtenga todos los enteros $100 \leq n \leq 400$ tales que $D(n)$ no es un álgebra de Boole.

Sabemos que dos álgebras de Boole finitas con el mismo cardinal son isomorfas ó equivalentes, lo cual se traduce en que ambas tienen el mismo diagrama de Hasse, salvo los nombres de los elementos. A continuación vamos a dibujar los diagramas de Hasse de las primeras álgebras de Boole finitas, omitiendo los nombres de los elementos. Para ello comenzamos cargando el paquete `graphs` de Maxima.

```
(%ixx) load(graphs)$
```

```
(%ixx) draw_graph(cube_graph(1));
```

El dibujo obtenido representa todas las álgebras de Boole de la forma $D(n)$, siendo n un número primo.

```
(%ixx) draw_graph(cube_graph(2));
```

El dibujo obtenido representa todas las álgebras de Boole de la forma $D(n)$, siendo $n = p_1 \cdot p_2$, producto de dos números primos distintos.

```
(%ixx) draw_graph(cube_graph(3));
```

El dibujo obtenido representa todas las álgebras de Boole de la forma $D(n)$, siendo $n = p_1 \cdot p_2 \cdot p_3$, producto de tres números primos distintos.

Lo anterior nos lleva al siguiente problema. Dadas las álgebras de Boole $D(m)$ y $D(n)$, ¿cómo podemos saber si éstas son o no isomorfas? La respuesta es muy fácil: $D(m)$ y $D(n)$ son isomorfas si y sólo si el número de primos que aparece en la factorización de m es igual que el número de primos que aparece en la factorización de n. Implementamos una función que lleva a cabo esta comprobación. Ojo, estamos suponiendo que $D(m)$ y $D(n)$ son ya álgebras de Boole.

```
(%ixx) iso(m,n):= is(length(ifactors(m)) = length(ifactors(n)));
```

Aplicamos la función a algunos ejemplos.

```
(%ixx) iso(51,155);
```

```
(%ixx) factor(51); factor(155);
```

```
(%ixx) iso(10,16278240338897);
```

```
(%ixx) factor(10); factor(16278240338897);
```

```
(%ixx) iso(210,3565271);
```

```
(%ixx) factor(210); factor(3565271);
```

En un álgebra de Boole finita sabemos que todo elemento distinto de $\mathbf{0}$ se escribe como suma de átomos y todo elemento distinto de $\mathbf{1}$ se escribe como producto de coátomos (es decir, complementos de átomos). En el caso de que nuestra álgebra de Boole venga dada como $D(n)$, sabemos que $\mathbf{0} = 1$ y $\mathbf{1} = n$. Además los átomos son precisamente los primos que dividen a n, mientras que los coátomos son todos los números de la forma $\frac{n}{p}$, siendo p un átomo.

Vamos a definir ahora un comando `atomos(n)` para obtener los átomos de $D(n)$, supuesto que éste es álgebra de Boole.

```
(%ixx) atomos(d):=block([lis,long],lis:ifactors(d),long:length(lis),
       setify(makelist(lis[i][1],i,1,long)))$
```

Probamos el comando definido con algunos ejemplos.

```
(%ixx) atomos(2);
```

```
(%ixx) atomos(6);
```

```
(%ixx) atomos(30);
```

```
(%ixx) atomos(892392217);
```

Si $D(n)$ es álgebra de Boole y $d \in D(n)$, con $d \neq 1$, este mismo comando nos permite obtener el conjunto de átomos $\{a_1, a_2, \ldots, a_k\} \subseteq D(n)$ tal que $d = a_1 + a_2 + \cdots + a_k$.

Comprobamos en primer lugar que $D(210)$ es un álgebra de Boole, y que 70 pertenece a la misma.

```
(%ixx) esAB(210);
```

```
(%ixx) elementp(70,divisors(210));
```

Los átomos asociados a 70 se obtienen con:

```
(%ixx) atomos(70);
```

Para comprobar que el supremo de tales átomos es 70, escribimos:

```
(%ixx) lcm(listify(%));
```

Es decir, hemos calculado el mínimo común múltiplo de todos los divisores primos de 70. Nótese que la salida del comando `atomos` es un conjunto, y hemos tenido que transformarlo en una lista para poder aplicarle el comando `lcm`.

Ejercicio 3. Compruebe que $D(1741209542339)$ es un álgebra de Boole, que $d = 1399667$ pertenece a la misma, y calcule los átomos que aparecen en la descomposición de d.

Ejercicio 4. Calcule el entero positivo más pequeño n que verifica simultáneamente las siguientes condiciones: $D(n)$ es un álgebra de Boole de cinco átomos y $10^5 \leq n$.
Realmente nos están preguntando por el menor número entero $\geq 10^5$ que se escribe como producto de cinco primos distintos. Para resolver este problema, intente aplicar el método comentado al principio de la práctica, es decir, comience generando un conjunto X suficientemente grande de números enteros $\geq 10^5$. Defina una función `f(n)` que devuelva `true` si n se escribe como producto de cinco primos distintos, y false en caso contrario. Por último obtenga el subconjunto de los elementos de X que verifican `f` y quédese con el menor de ellos.

Análogamente al comando `atomos`, ahora definimos el comando `coatomos(n)` que nos permitirá obtener los coátomos de $D(n)$, supuesto que éste es álgebra de Boole.

```
(%ixx) coatomos(n):=block([la],la:listify(atomos(n)),
       setify(makelist(quotient(n,la[k]),k,1,length(la))))$
```

Probamos este comando con algunos ejemplos.

```
(%ixx) coatomos(30);
```

```
(%ixx) coatomos(210);
```

```
(%ixx) coatomos(892392217);
```

```
(%ixx) coatomos(6);
```

Según las definiciones, el producto (es decir, el máximo común divisor) de todos los elementos pertenecientes al conjunto `coatomos(n)` ha de ser igual a 1.

```
(%ixx) X:coatomos(30);
```

Como hemos visto antes, la salida obtenida es $\{6, 10, 15\}$. Una peculiaridad de Maxima, y al contrario de lo que ya hemos visto que ocurre con la función `lcm`, es que la función `gcd`, que calcula el máximo común divisor, sólo se puede aplicar a dos números. Obtenemos el máximo común divisor de los números 6, 10, 15 como sigue.

```
(%ixx) gcd(gcd(6,10),15);
```

Si $D(n)$ es un álgebra de Boole y $d \in D(n)$, con $d \neq n$, podemos obtener el conjunto de coátomos $\{c_1, c_2, \ldots, c_r\} \subseteq D(n)$ tal que $d = c_1 \cdot c_2 \cdots c_r$ mediante el comando coatomoss(d,n) que definimos seguidamente.

```
(%ixx) coatomoss(d,n):=block([lis,long],lis:ifactors(quotient(n,d)),
        long:length(lis),setify(makelist(quotient(n,lis[i][1]),i,1,long)))$
```

Probamos este comando con algunos ejemplos.

```
(%ixx) coatomoss(2,30);
```

Es decir, $2 = 6 \wedge 10 = mcd(6, 10)$.

```
(%ixx) gcd(6,10);
(%ixx) coatomoss(1,210);
(%ixx) coatomoss(3,210);
```

Como sabemos de teoría, toda álgebra de Boole finita también puede ser dada de la forma $\mathcal{P}(X)$, siendo X un conjunto finito. En Maxima existe el comando `powerset(X)` que calcula el conjunto potencia de un conjunto `X`.

```
(%ixx) X:{1,2,3,4,5};
```

Hemos definido un conjunto X cuyos elementos son 1,2,3,4,5. Su conjunto potencia es:

```
(%ixx) powerset(X);
```

Ahora las operaciones $\vee$, $\wedge$ y complemento se calculan con los comandos `union`, `intersection` y `setdifference`. Veamos unos ejemplos.

```
(%ixx) A:{2,4,5}; B:{1,2,5};
```

Hemos definido dos elementos A y B de $\mathcal{P}(X)$, es decir, dos subconjuntos del conjunto X. La suma de A y B se obtiene con

```
(%ixx) union(A,B);
```

El producto de A y B, con

```
(%ixx) intersection(A,B);
```

El complementario de A se calcula mediante

```
(%ixx) setdifference(X,A);
```

y el complementario de B mediante

```
(%ixx) setdifference(X,B);
```

De forma análoga a lo visto para $D(n)$, se pueden obtener los átomos y los coátomos para las álgebras de Boole representadas de la forma $\mathcal{P}(X)$.

2. Optimización de funciones booleanas

Para realizar con Maxima una minimización de funciones booleanas siguiendo el método tabular de **Quine-McCluskey**, debemos acudir al conjunto de paquetes "Discrete", el cual contiene el paquete *boolmin.lisp*, adecuado para ello. Para su correcto funcionamiento, previamente cargamos los paquetes (desde el menú *Archivo* de la aplicación, salvo que los tengamos en *contrib*):

```
(%ixx) load(logic_ops)$
(%ixx) load(logic)$
(%ixx) load(boolmin)$
```

Y ahora ya podemos pedirle que minimice una función. La notación se hará con los operadores lógicos *not*, *and* y *or*. Por ejemplo, la minimización más simple vista en clase :

```
(%ixx) boolean_minimize((x and y) or (not x and y) or (x and not y));
```

que da como resultado $x + y$, o en notación lógica $x \vee y$.

Tambien podemos probar con alguna de las minimizaciónes vistas en clase:

```
(%ixx) boolean_minimize((not x and not y and not z ) or (x and not y and not z) or
       (not x and y and not z) or (not x and not y and z) or (x and y and not z));
```

Asimismo, en el siguiente enlace, podemos encontrar una aplicación muy simple que minimiza funciones booleanas.

http://www.mathcs.bethel.edu/~gossett/DiscreteMathWithProof/QuineMcCluskey.html

En la casilla superior introducimos el número de variables n, que por defecto es 2. Entonces aparece una tabla con 2^n filas en las cuales marcamos los minterms que componen la función que queremos minimizar. De forma automática aparece en la columna de la derecha la expresión simplificada.

Ejercicio 7 Obtenga una expresión mínima como suma de productos para la función booleana

$$f(x_1, \ldots, x_7) = \sum m(0, 2, 14, 15, 42, 43, 44, 45, 90, 91, 94, 100, 101, 111, 103, 123, 124, 127).$$