Tarea 1

Profesor: Pablo Barceló - **Auxiliares:** Bernardo Subercaseaux, Javier Oliva
Ayudantes: Joaquin Cruz, Heinich Porro, Lucas Torrealba, Florencia Yañez
Alumno: Sebastián Sepúlveda A.

Soluciones

P1 En lógica proposicional, un literal es una variable o la negación de una variable. Es decir,p y $\neg q$ son literales, pero no $q \vee \neg p$. Una cláusula es la disyunción de literales. Es decir,$(p \vee q \vee \neg r)$ y $(\neg p \vee p \vee \neg q)$ son cláusulas. Mientras que $(p \vee q \wedge r)$ no lo es. Decimos que una fórmula de la lógica proposicional está en CNF si se escribe como conjunción de cláusulas, es decir como $C_1 \wedge C_2 \wedge C_3 \wedge \ldots \wedge C_n$ donde cada C_i es una cláusula.

P1.1 Pruebe que toda fórmula de la lógica proposicional se puede escribir en CNF.

Para demostrar que toda fórmula de la lógica proposicional se puede demostrar en CNF es suficiente demostrar que CNF es funcionalmente completo.

Para esto, notamos que cada formula CNF se puede escribir como conjunción de cláusulas, y que cada cláusula es la disyunción de al menos 2 literales. Luego, por definición dos literales son p o $\neg q$, por lo que cada cláusula está representada por $(\neg\ ,\ \vee)$. Este resultado implica que CNF está representado por $\wedge\ ,\ \vee\ ,\ \neg$, que por resultado visto en clases es funcionalmente completo.

P1.2 Pruebe que para toda fórmula en CNF de n cláusulas existe una valuación donde al menos $n/2$ cláusulas se satisfacen.

La intuición nos señala que la propuesta es cierta. En efecto, en el caso minimo tenemos dos cláusulas de la forma $C_1 = (p \vee \neg q) \wedge C_2 = (r \vee \neg s)$, donde el peor caso es cuando existe valuación σ_1 tal que $\sigma_1(C_1) = 0 \wedge \sigma_1(C_2) = 0$. Esto nos señala que la valuación es tal que a todos los literales que tienen cada cláusula le asigna valor 0. Sin embargo, sabemos que existen otros σ_2 tal que $\sigma_2(C_1) = 1 \wedge \sigma_2(C_2) = 1$, por definición de los literales de que existe la negación de su valor, y por definición de valuación, pues le asigna *uno* a al menos un literal en cada cláusula. El resultado basta para señalar que al menos $n/2$ cláusulas se satisfacen, pero no es la cota minima, pues claramente si pudimos encontrar σ_2 que hiciera cierto a todas las cláusulas, entonces podemos encontrar σ_3 que haga cierta a al menos 1 cláusula, y mantenga la condición de σ_1 para la otra cláusula. Finalmente, para este caso en particular tenemos que, para toda fórmula en CNF con $n = 2$ cláusulas la minima cota que satisface a todas las cláusulas es al menos $n/2$

Consideramos cada cláusula por separado, es decir, n cláusulas C_i con $n \in \mathbb{N}$ arbitrario, de tamaño cualquiera K_i con $i \in [1, \ldots, n]$. Sabemos que para algún σ cualquiera las cláusulas pueden tomar distintos valores de verdad 0 ó 1. En cada cláusula donde la valuación da 0, todos los literales evaluados por σ deben ser 0, por definición de cláusula. El resultado de los C_i tal que $\sigma(C_i) = 0$ lo consideramos como una sola cláusula. Luego para las cláusulas que dan 1, sabemos que $\exists$ al menos un literal que permite satisfacer todas aquellas cláusulas. Estas dos familias de cláusulas, las que entregan resultados 1 y las que entregan resultado 0 las vamos a separar en dos grupos.

$$\begin{aligned} A = \{C_i,\ \ i \in [1,k] \quad &\big|\ \forall p \in C_i,\ \ \sigma(p) = 0 \Rightarrow \sigma(C_i) = 0\} \\ \wedge\ B = \{C_i,\ \ i \in [k+1,n] \ &\big|\ \exists p \in C_i,\ \ \sigma(p) = 1 \Rightarrow \sigma(C_i) = 1\} \end{aligned}$$

Luego, en el caso de que la valuación de la fórmula en CNF dé que hay más $0's$ que $1's$, se tiene que $k > n/2$. Pero dado a que todo literal en C_i implica $C_i = 0$, con $i \in [1, k]$, basta considerar una nueva valuación $\widetilde{\sigma} = \neg\sigma \Rightarrow \exists p \in C_i, \ \sigma(p) = 0 \wedge \sigma C_i = 1$ donde se obtiene que a lo menos $k > n/2$ cláusulas se satisfacen, en particular $n/2$ cláusulas. Como lo hicimos para un $n \in \mathbb{N}$ arbitrario, la propiedad se tiene para toda fórmula en CNF con n cláusulas.

P1.3 Pruebe que el resultado anterior no se puede mejorar. Es decir, no existe ningún $r > 1/2$ tal que para toda fórmula en CNF de n cláusulas exista una valuación donde al menos rn cláusulas se satisfacen.

Para demostrar esta pregunta lo haremos por contradicción. Supongamos que $\exists r > 1/2$ tal que para toda fórmula en CNF de n cláusulas existe una valuación donde al menos rn cláusulas se satisfacen. Consideremos la familia de las cláusulas de largo 4. Dado a que $r > 1/2$, por ejemplo $r = 3/4$, toda fórmula en CNF debe tener como cota minima de cláusulas que se satisfacen a $r > 1/2$, pero si por ejemplo consideramos la formula CNF con 4 clausulas: $C_1 = p \vee \neg q$, $C_2 = \neg p \vee q$, $C_2 = p \vee \neg p$,$C_4 = p \vee \neg p$, y la valuación que hace $p = 0 \wedge q = 0$ se tiene que la mitad se satisfacen. Por lo tanto, no puede existir $r > 1/2$ tal que para toda fórmula en CNF con n cláusulas se satisfacen rn

P2 Decimos que un número se escribe de forma factorial como:

$$3010! = 3 \cdot 3! + 0 \cdot 2! + 1 \cdot 1! + 0 \cdot 0!$$

Más formalmente el valor de un número escrito de forma factorial $(a_k a_{k-1} a_{k-2} \dots a_0)_!$ es igual $\sum_{i=0}^{k} a_i \cdot i!$, donde $0 \leqslant a_i \leqslant i$.

P2.1 Pruebe que todo natural se puede escribir de forma factorial, y que además, esa expresión es única (no hay dos formas distintas de escribir un número en forma factorial).

PDQ la propiedad se cumple para todo $n \in \mathbb{N}$.

Nuestro caso base será $(10)_! = 1$, donde se cumple que $\sum_{i=0}^{1} 1 \cdot i! = 1$ y $0 \leqslant 1 \leqslant 1$.

La estrategia que se ocupará para demostrarlo para todo n, será encontrar el menor factorial más cercano a n, es decir, un $k!$, con $k \in \mathbb{N}$ donde $k! \leqslant n < (k+1)! \ \ \forall \ n \geqslant 1$. Conociendo este k conocemos el largo del número en forma factorial, pues como máximo $k \cdot k! + C = n$, con $C = cte$ y con $k \cdot k! + C \neq (k+1)!$ por la condición que nos dimos al encontrar k.

En caso contrario, si permitimos que $k \cdot k! + C = (k+1)!$ contradecimos el hecho de que n tenga un único k que sea el menor factorial a este número, pues los naturales son un conjunto de orden total. Luego tenemos que k es único, y por tanto el largo es único.

Al conocer k, sabemos que $(n - k!) \geqslant 0$ y $(n - k!) \in \mathbb{N} \cup \{0\}$ a este resultado lo llamaremos $(n - k!) = n_2$ donde existen 3 casos:

- $k! < n_2$ entonces $k! < n - k! \Leftrightarrow 2 \cdot k! < n$, en cuyo caso $k!$ ya no es el menor natural cercano a n y lo reemplazamos por $2 \cdot k!$
- $k! > n_2$ entonces sabemos que $\exists k_2 \leqslant k - 1$ tq $k_2! \leqslant n_2$, que definiría al siguiente digito en el arreglo
- $n_2 = 0$ entonces encontramos la forma de representar el número en forma factorial.

Como nuestro método es recursivo y se cumple para cada $n \in \mathbb{N}$ concluimos.

P2.2 Programe una función `baseFact` que recibe un natural n y retorna un vector m que representa al número en forma factorial, es decir $m_! = n_{10}$.

Para la redacción del código se consideró la entrada de un $n > 0$ con $n \in \mathbb{N}$, y se ocupó el razonamiento descrito en la pregunta anterior. Primero, obviamos el hecho que $[0]_! = 0$ lo cual escapa de la definición. Se ocupó dos ciclos `while`. El primero fue para construir un arreglo con k elementos, tal que $k! \leq n$ que se detiene cuando existe un factorial(k) . El siguiente `while` se ocupó para generar la forma factorial de derecha a izquierda, y cubrir los casos descritos anteriormente, resolviendo también los sub-casos dentro del arreglo.

Para conocer el factorial de un número se utilizó el módulo `math` de python.

P2.3 Entregue un archivo que al ser ejecutado reciba por entrada estándar un número natural n, e imprima el resultado de la función anterior aplicada sobre n, separándolos distintos dígitos con un signo '#'

En el proceso de interacción de usuario, se crea una función `printBaseFact()` que con la función `*<list>` permite imprimir en una linea con un parámetro `sep=`, en el que se indica la separación deseada.

P3.1 Pruebe que todo natural se puede escribir en forma de Fibonacci sin que en su representación aparezcan dos unos consecutivos. Pruebe además que bajo esta condición la representaciónes única.

En este problema se ocupó el mismo razonamiento del ejercicio anterior. Sea $n \in \mathbb{N}$ $\{0\}$ y $f_k \leq n$ con $k \geq 2 \ \wedge \ k \in \mathbb{N}$. Nuestro caso base será para $n = 2$ que con $k = 3$ se cumple que tiene forma de Fibonacci: $[1, 0]$, pues $f_3 + 0 \cdot f_2 = 2$.

Hipótesis inductiva: Supongamos que se cumple para n, es decir n tiene representación factorial:

$$n \equiv (a_k a_{k-1} \dots a_2)_f \ \text{ con } \ a \in \{0,1\} \ \wedge \ \sum_{i=0}^{k} a_i f_i = n$$

PDQ para $n+1$ se cumple, con $k > 3$. En efecto sabemos que $f_k \leq n < n+1$. Tenemos dos casos: $n = f_k \vee n > f_k$

- $n = f_k$: sabemos que $\sum_{i=0}^{k} f_i a_i = f_k = n < n+1$ luego basta considerar $f_k + f_2 = f_k + 1$ para concluir.
- $n > f_k$: tenemos dos casos también: cuando n es la suma de los números de Fibonacci con distancia 1 y el caso de ser distancia mayor que 1
  - En el primer caso, sea $j \in \left[0, \dfrac{k-2}{2}\right]$ tenemos que:

$$\begin{aligned} n = \sum_{i=0}^{k} f_i a_i &= f_k + f_{k-2} + \dots + f_{k-2j} + \dots + f_2 \\ &\Leftrightarrow n = f_{k-1} + f_{k-2} + f_{k-3} + f_{k-4} + \dots + f_1 + f_0 \qquad \text{con } \sum_{i=0}^{n} f_i = f_{n+2} - 1 \\ &\Leftrightarrow n = f_{(k-1)+2} - 1 \\ &\Leftrightarrow n = f_{k+1} - 1 \Leftrightarrow \ n + 1 = f_{k+1} \end{aligned}$$

Luego para demostrar la unicidad lo haremos por contradicción supongamos que n tiene dos representaciones, es decir, posee distinto largo. Eso implica que $n = \sum_{i=0}^{k} a_i f_i \wedge n = \sum_{i=0}^{l} a_i f_i$ con $k \neq l$. Por el

paso anterior, sabemos que $n = f_{k+1} - 1 = f_{l+1} - 1 \rightarrow\ f_k = f_l$ lo cual es una contradicción por hipótesis y por definición de la sucesión de Fibonacci.

- Para el segundo caso, dado a que la distancia entre cada numero de Fibonacci de la suma no es necesariamente intercalada, y es mayor que 1 al menos en 1, y estamos considerando calcular el sucesor de n, entonces solo se reduce a desplazar el ultimo dígito de la derecha una unidad hacia la izquierda, por ejemplo $n \equiv [1,0,0,0,1] \rightarrow n+1 = [1,0,0,1,0]$ pues los otros casos ya los cubrimos, o son numeros mayores al sucesor.

Finalmente se concluye que para todo natural $n > 1$ se cumple la forma de Fibonacci.

`Nota: la unicidad también se obtiene por la unicidad de la suma en los naturales. Dado a que la sucesión de Fibonacci es subconjunto de los naturales, entonces la suma entre los términos de la sucesión también es única.`

P3.2 Programe una función `baseFib` que recibe un natural n y retorna un vector m que representa al número en forma de Fibonacci sin unos consecutivos.

Para la redacción del código se consideró la entrada de algún natural $n > 2$ y se siguió el mismo razonamiento del problema anterior. Se creó una función auxiliar `sucFib(n)` que entrega la n-ésimo término de la sucesión. Luego se hicieron 2 ciclos `while` tal que uno genera el tamaño del arreglo, con el menor número de Fibonacci más cercano al número entregado y el otro suma tantos $1's$ sean necesarios para que la suma sea el número que se ingresó.

P3.3 Se hizo similar a la pregunta 2.