Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Tarea 2

26 de agosto de 2024

2º semestre 2024 - Profesores P. Bahamondes - D. Bustamante - M. Romero

---

Requisitos

- La tarea es individual. Los casos de copia serán sancionados con la reprobación del curso con nota 1,1.
- **Entrega**: Hasta las 23:59 del 04 de septiembre a través del buzón habilitado en el sitio del curso (Canvas).
  - Esta tarea debe ser hecha completamente en LaTeX. Tareas hechas a mano o en otro procesador de texto **no serán corregidas**.
  - Debe usar el template LaTeX publicado en la página del curso.
  - Cada solución de cada problema debe comenzar en una nueva hoja. ***Hint:*** Utilice `\newpage`
  - Los archivos que debe entregar son el archivo `PDF` correspondiente a su solución con nombre `numalumno.pdf`, junto con un `zip` con nombre `numalumno.zip`, conteniendo el archivo `numalumno.tex` que compila su tarea. Si su código hace referencia a otros archivos, debe incluirlos también. En caso de entregar el bonus de código, este debe incluirse por separado en único archivo con nombre `numalumno.py`.
- El no cumplimiento de alguna de las reglas se penalizará con un descuento de 0.5 en la nota final (acumulables).
- La nota final de la tarea es la nota obtenida sin el bonus opcional (e) de la Pregunta 1, más 1 punto, en caso de haber entregado el bonus y tenerlo correcto. En caso de obtener una nota mayor a 7.0, **no** se acumularán décimas.
- No se aceptarán tareas atrasadas (salvo que utilice su cupón `#problemaexcepcional`).
- Si tiene alguna duda, el foro de Github (issues) es el lugar oficial para realizarla.

Pregunta 1

Sea Σ un conjunto finito no vacío de símbolos. Se define el conjunto Σ^* de las palabras sobre Σ, como el menor conjunto que cumple las siguientes reglas:

- $\varepsilon \in \Sigma^*$, donde ε corresponde a la palabra vacía que no contiene símbolos.
- Si $w \in \Sigma^*$ y $a \in \Sigma$, entonces $wa \in \Sigma^*$.

A modo de ejemplo, si $\Sigma = \{a, b, c\}$, algunas posibles palabras en Σ^* serían:

$$\varepsilon \qquad b \qquad acc \qquad cbabb$$

Notar que por comodidad, omitimos la palabra vacía ε cuando no es necesaria, es decir, escribimos a, acc y $cbabb$, en vez de εa, εacc y $\varepsilon cbabb$, respectivamente.

(a) Sea $a \in \Sigma$. Defina inductivamente la función $\#_a : \Sigma^* \to \mathbb{N}$, que a cada palabra le asigna la cantidad de ocurrencias del símbolo a en la palabra. Algunos ejemplos:

$$\#_a(b) = 0 \qquad \#_b(b) = 1 \qquad \#_c(acc) = 2 \qquad \#_b(cbabb) = 3$$

(b) Sean $a, b \in \Sigma$. Defina inductivamente la función $r_{a\to b} : \Sigma^* \to \Sigma^*$, que a cada palabra le asigna la palabra resultante de reemplazar cada símbolo a por el símbolo b. Algunos ejemplos:

$$r_{c\to b}(acc) = abb \qquad r_{b\to a}(acc) = acc \qquad r_{b\to a}(cbabb) = caaaa \qquad r_{c\to c}(cbabb) = cbabb$$

(c) Sean $a, b \in \Sigma$ tal que $a \neq b$. Demuestre usando inducción estructural que para toda palabra $w \in \Sigma^*$ se cumple:

$$\#_b(r_{a\to b}(w)) = \#_a(w) + \#_b(w)$$

(d) Sean $a, b \in \Sigma$ tal que $a \neq b$. Demuestre usando inducción estructural que para toda palabra $w \in \Sigma^*$ se cumple:

$$r_{b\to a}(r_{a\to b}(w)) = w \implies \#_b(w) = 0$$

La siguiente pregunta es opcional y ofrece un bonus de 1 punto sobre la nota final de la tarea (ver instrucciones en la primera página).

(e) Utilizando su definición inductiva de la función $r_{a \to b}$, escriba en `Python` un programa recursivo `reemplazar_simbolo(w, a, b)`, que dada una palabra w, y dos símbolos a y b, retorna la palabra correspondiente a $r_{a \to b}(w)$.

Importante: No puede usar ninguna función de `Python` que haga el reemplazo directamente. Debe respetar la definición inductiva de Σ^* y la definición inductiva de $r_{a \to b}$.

Condiciones de entrega: Un único archivo de nombre `numero_alumno.py` tal que su solución se pruebe haciendo un llamado a la función `reemplazar_simbolo(w, a, b)` en dicho archivo.

Ejemplos de ejecución:

`reemplazar_simbolo('acc', 'c', 'b')`	retorna `'abb'`
`reemplazar_simbolo('acc', 'b', 'a')`	retorna `'acc'`
`reemplazar_simbolo('cbabb', 'b', 'a')`	retorna `'caaaa'`
`reemplazar_simbolo('cbabb', 'c', 'c')`	retorna `'cbabb'`

Solución

(a) Sea $a \in \Sigma$. Podemos definir inductivamente la función $\#_a : \Sigma^* \to \mathbb{N}$ como sigue:

1. $\#_a(\varepsilon) = 0$.
2. Si $w \in \Sigma^*$ y $b \in \Sigma$, entonces distinguimos dos casos:
   - Si $b = a$, definimos $\#_a(wb) = \#_a(w) + 1$.
   - Si $b \neq a$, definimos $\#_a(wb) = \#_a(w)$.

Puntaje:

- 0.2 ptos por definir la función correctamente en el caso base.
- 0.8 ptos por definir la función correctamente en el caso inductivo.

Descuentos y puntajes parciales a criterio del corrector.

(b) Sean $a, b \in \Sigma$. Podemos definir inductivamente la función $r_{a \to b} : \Sigma^* \to \Sigma^*$ como sigue:

1. $r_{a \to b}(\varepsilon) = \varepsilon$.
2. Si $w \in \Sigma^*$ y $c \in \Sigma$, entonces distinguimos dos casos:
   - Si $c = a$, definimos $r_{a \to b}(wc) = r_{a \to b}(w)b$.
   - Si $c \neq a$, definimos $r_{a \to b}(wc) = r_{a \to b}(w)c$.

Puntaje:

- 0.2 ptos por definir la función correctamente en el caso base.
- 0.8 ptos por definir la función correctamente en el caso inductivo.

Descuentos y puntajes parciales a criterio del corrector.

(c) Sean $a, b \in \Sigma$ tal que $a \neq b$. Demostremos por inducción estructural que para toda palabra $w \in \Sigma^*$ se cumple:

$$\#_b(r_{a\to b}(w)) = \#_a(w) + \#_b(w)$$

BI: Aplicando las definiciones de $\#_a$, $\#_b$ y $r_{a\to b}$ obtenemos:

$$\#_b(r_{a\to b}(\varepsilon)) = \#_b(\varepsilon) = 0 = 0 + 0 = \#_a(\varepsilon) + \#_b(\varepsilon)$$

HI: Sea $w \in \Sigma^*$ tal que $\#_b(r_{a\to b}(w)) = \#_a(w) + \#_b(w)$.

TI: Sea $c \in \Sigma$ arbitrario. Debemos demostrar que:

$$\#_b(r_{a\to b}(wc)) = \#_a(wc) + \#_b(wc)$$

Distinguimos tres casos:

- Supongamos que $c = a$. Aplicando las definiciones de $\#_a$, $\#_b$ y $r_{a\to b}$ y la HI, obtenemos:

$$\begin{aligned}
\#_b(r_{a\to b}(wc)) &= \#_b(r_{a\to b}(w)b) && (c = a \text{ y def } r_{a\to b})\\
&= \#_b(r_{a\to b}(w)) + 1 && (\text{def } \#_b)\\
&= \#_a(w) + \#_b(w) + 1 && (\text{HI})\\
&= \#_a(w) + 1 + \#_b(wc) && (c \neq b \text{ y def } \#_b)\\
&= \#_a(wc) + \#_b(wc) && (c = a \text{ y def } \#_a)
\end{aligned}$$

- Supongamos que $c = b$. Aplicando las definiciones de $\#_a$, $\#_b$ y $r_{a\to b}$ y la HI, obtenemos:

$$\begin{aligned}
\#_b(r_{a\to b}(wc)) &= \#_b(r_{a\to b}(w)b) && (c \neq a \text{ y def } r_{a\to b})\\
&= \#_b(r_{a\to b}(w)) + 1 && (\text{def } \#_b)\\
&= \#_a(w) + \#_b(w) + 1 && (\text{HI})\\
&= \#_a(wc) + \#_b(w) + 1 && (c \neq a \text{ y def } \#_a)\\
&= \#_a(wc) + \#_b(wc) && (c = b \text{ y def } \#_b)
\end{aligned}$$

- Supongamos que $c \notin \{a, b\}$. Aplicando las definiciones de $\#_a$, $\#_b$ y $r_{a\to b}$ y la HI, obtenemos:

$$\begin{aligned}
\#_b(r_{a\to b}(wc)) &= \#_b(r_{a\to b}(w)c) && (c \neq a \text{ y def } r_{a\to b})\\
&= \#_b(r_{a\to b}(w)) && (c \neq b \text{ y def } \#_b)\\
&= \#_a(w) + \#_b(w) && \text{(HI)}\\
&= \#_a(wc) + \#_b(w) && (c \neq a \text{ y def } \#_a)\\
&= \#_a(wc) + \#_b(wc) && (c \neq b \text{ y def } \#_b)
\end{aligned}$$

Puntaje:

- 0.2 ptos por el caso base.
- 0.3 ptos por plantear correctamente la hipotesis inductiva.
- 1.5 ptos por demostrar correctamente la tesis inductiva.

Descuentos y puntajes parciales a criterio del corrector.

(d) Sean $a, b \in \Sigma$ tal que $a \neq b$. Demostremos por inducción estructural que para toda palabra $w \in \Sigma^*$ se cumple:

$$r_{b\to a}(r_{a\to b}(w)) = w \implies \#_b(w) = 0$$

BI: Aplicando las definiciones de $r_{a\to b}$, $r_{b\to a}$ y $\#_b$, tenemos que $r_{b\to a}(r_{a\to b}(\varepsilon)) = \varepsilon$ y $\#_b(\varepsilon) = 0$. Luego, se cumple que $r_{b\to a}(r_{a\to b}(\varepsilon)) = \varepsilon \implies \#_b(\varepsilon) = 0$.

HI: Sea $w \in \Sigma^*$ tal que $r_{b\to a}(r_{a\to b}(w)) = w \implies \#_b(w) = 0$.

TI: Sea $c \in \Sigma$ arbitrario. Debemos demostrar que:

$$r_{b\to a}(r_{a\to b}(wc)) = wc \implies \#_b(wc) = 0$$

Distinguimos tres casos:

- Supongamos que $c = a$. Asumamos que $r_{b\to a}(r_{a\to b}(wc)) = wc$. Debemos demostrar que $\#_b(wc) = 0$. Por definición de $r_{a\to b}$ y $r_{b\to a}$, tenemos que

  $$r_{b\to a}(r_{a\to b}(wc)) = r_{b\to a}(r_{a\to b}(w)b) = r_{b\to a}(r_{a\to b}(w))a$$

  El supuesto $r_{b\to a}(r_{a\to b}(wc)) = wc$ implica que $r_{b\to a}(r_{a\to b}(w))a = wa$. A su vez, esto implica que $r_{b\to a}(r_{a\to b}(w)) = w$. Por HI, sigue que $\#_b(w) = 0$. Como $c \neq b$, y por definición de $\#_b$, concluimos que $\#_b(wc) = \#_b(w) = 0$.

- Supongamos que $c = b$. En este caso, la implicancia $r_{b\to a}(r_{a\to b}(wc)) = wc \implies \#_b(wc) = 0$ se cumple trivialmente, ya que $r_{b\to a}(r_{a\to b}(wc)) = wc$ es falso. Veamos esto último. Por definición de $r_{a\to b}$ y $r_{b\to a}$, tenemos que $r_{b\to a}(r_{a\to b}(wc)) = r_{b\to a}(r_{a\to b}(w)b) = r_{b\to a}(r_{a\to b}(w))a$. Se cumple que

$$r_{b\to a}(r_{a\to b}(wc)) = r_{b\to a}(r_{a\to b}(w))a \neq wb = wc$$

  ya que $a \neq b$.

- Supongamos que $c \notin \{a, b\}$. Asumamos que $r_{b\to a}(r_{a\to b}(wc)) = wc$. Debemos demostrar que $\#_b(wc) = 0$. Por definición de $r_{a\to b}$ y $r_{b\to a}$, tenemos que

$$r_{b\to a}(r_{a\to b}(wc)) = r_{b\to a}(r_{a\to b}(w)c) = r_{b\to a}(r_{a\to b}(w))c$$

  El supuesto $r_{b\to a}(r_{a\to b}(wc)) = wc$ implica que $r_{b\to a}(r_{a\to b}(w))c = wc$. A su vez, esto implica que $r_{b\to a}(r_{a\to b}(w)) = w$. Por HI, sigue que $\#_b(w) = 0$. Como $c \neq b$, y por definición de $\#_b$, concluimos que $\#_b(wc) = \#_b(w) = 0$.

Puntaje:

- 0.2 ptos por el caso base.
- 0.3 ptos por plantear correctamente la hipotesis inductiva.
- 1.5 ptos por demostrar correctamente la tesis inductiva.

Descuentos y puntajes parciales a criterio del corrector.

(e) La función `reemplazar_simbolo(w, a, b)` toma como entrada un string w y dos caracteres a y b, y retorna la palabra correspondiente a $r_{a\to b}(w)$, es decir, reemplaza a por b.

El algoritmo sigue los siguientes pasos:

1. Si w es el string vacío, entonces retornamos w.
2. Si w no es el string vacío, y w es de la forma $w = xc$, donde c es el último caracter, hacemos lo siguiente:
   a) Si $c = a$, llamamos recursivamente a `reemplazar_simbolo(x, a, b)` obteniendo el string y. Retornamos el string yb.
   b) Si $c \neq a$, llamamos recursivamente a `reemplazar_simbolo(x, a, b)` obteniendo el string y. Retornamos el string yc.

A continuación un posible código:

```
def reemplazar_simbolo(w,a,b):
      if w == '':
           return w
```

```
if w[-1] == a:
    return reemplazar_simbolo(w[:-1],a,b) + b
else:
    return reemplazar_simbolo(w[:-1],a,b) + w[-1]
```

Puntaje:

- Si no se aplica recursión, o no se usa la definición inductiva de Σ^* ni de la función $r_{a\to b}$, esta pregunta no recibe puntaje.

Pregunta 2

Suponga que tenemos un conjunto $V = \{1, \ldots, n\}$ de n personas. Algunas parejas de personas son compatibles y otras no. Estas relaciones de compatibilidad están descritas por un conjunto E de parejas de personas compatibles. Es decir, $(i, j) \in E$ si y sólo si las personas i y j son compatibles. Un *núcleo* para V y E es un subconjunto C de personas de V que son compatible entre ellas, es decir, tal que para todo par (i, j) de personas en C se cumple que $(i, j) \in E$.

A modo de ejemplo, suponga que nuestro conjunto de personas es $V = \{1, 2, 3, 4\}$ y nuestro conjunto de compatibilidades es $E = \{(1, 2), (2, 3), (1, 3), (1, 4)\}$. Tenemos que $C = \{1, 2, 3\}$ es un núcleo para V y E de 3 personas, ya que tenemos las compatibilidades $(1, 2)$, $(2, 3)$ y $(1, 3)$. Por otra parte, si escogemos $C = \{1, 3, 4\}$, no obtenemos un núcleo ya que 3 y 4 no son compatibles.

Dado un conjunto V de n personas, un conjunto de compatibilidades E y un parámetro $k \leq n$, nuestra misión es encontrar un núcleo con k personas para V y E. Por supuesto, queremos utilizar la lógica proposicional para resolver este problema. Escriba una fórmula en lógica proposicional φ tal que:

Existe un núcleo con k personas para V y E si y sólo si φ es satisfacible.

Debe demostrar que su fórmula φ es correcta, es decir, que cumple la propiedad enunciada arriba.

Para definir φ, **debe** utilizar las siguientes variables proposicionales:

- Variables p_{ij}, donde $1 \leq i, j \leq n$, que expresan que i y j son compatibles.
- Variables x_{hi} donde $1 \leq h \leq k$ e $1 \leq i \leq n$, que expresan que la persona i es la h-ésima persona del núcleo.

Solución

Tenemos dado un conjunto de personas $V = \{1, \ldots, n\}$ de tamaño n, las compatibilidades $E \subseteq V \times V$, y un parámetro $k \geq n$. Construiremos una fórmula proposicional φ tal que:

Existe un núcleo para (V, E) de tamaño k si y sólo si φ es satisfacible.

La fórmula φ tiene dos tipos de variables proposicionales:

- Variables p_{ij}, donde $1 \leq i, j \leq n$ que expresan que i y j son compatibles.
- Variables x_{hi}, donde $1 \leq h \leq k$ e $1 \leq i \leq n$, que expresan que la persona i es la h-ésima persona del núcleo.

Para definir φ definimos las siguientes fórmulas auxiliares:

- Una fórmula ψ que inicializa las variables p_{ij}:

$$\psi = \bigwedge_{(i,j)\in E} p_{ij} \wedge \bigwedge_{(i,j)\notin E} \neg p_{ij}$$

  Observación: En estricto rigor, solo es necesario el caso negativo de las variables p_{ij}, pues en la fórmula φ_3 de más abajo, aparecen en el consecuente de la implicancia y por lo tanto solo en el caso negativo son restrictivas en la satisfactibilidad de la fórmula. En consecuencia, la siguiente alternativa de inicialización también es correcta y aceptada como solución:

$$\psi = \bigwedge_{(i,j)\notin E} \neg p_{ij}$$

- Una fórmula φ_1 que expresa que cada posición $1 \leq h \leq k$ tiene una y solo una persona asignada en el núcleo:

$$\varphi_1 = \bigwedge_{h=1}^{k} \left(\left(\bigvee_{i=1}^{n} x_{hi} \right) \wedge \left(\bigwedge_{i=1}^{n} \left(x_{hi} \rightarrow \bigwedge_{\substack{j=1\\ j\neq i}}^{n} \neg x_{hj} \right) \right) \right)$$

  Otra alternativa:

$$\varphi_1 = \bigwedge_{h=1}^{k} \bigvee_{i=1}^{n} \left(x_{hi} \wedge \bigwedge_{\substack{j=1\\ j\neq i}}^{n} \neg x_{hj} \right)$$

- Una fórmula φ_2 que expresa que a posiciones distintas $h, g \in \{1, \ldots, k\}$ se le asignan personas distintas en el núcleo. Esto nos asegura que el núcleo tiene tamaño k.

$$\varphi_2 = \bigwedge_{h=1}^{k} \bigwedge_{i=1}^{n} \left(x_{hi} \rightarrow \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \neg x_{gi} \right)$$

  Otra alternativa:

$$\varphi_2 = \bigwedge_{i=1}^{n} \bigwedge_{h=1}^{k} \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \left(\neg (x_{hi} \wedge x_{gi}) \right)$$

- Una fórmula φ_3 que expresa que todas las personas asignadas al núcleo son compatibles:

$$\varphi_3 = \bigwedge_{h=1}^{k} \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=1\\ i<j}}^{n} \left(\left(x_{hi} \wedge x_{gj} \right) \rightarrow p_{ij} \right)$$

  Otra alternativa:

$$\varphi_3 = \bigwedge_{h=1}^{k} \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=1\\ i<j}}^{n} \left(\neg p_{ij} \rightarrow (\neg x_{hi} \vee \neg x_{gj}) \right)$$

Dado que no está completamente especificado lo que es un par ordenado a estas alturas del curso ni especificamos que $i < j$, las siguientes alternativas también se consideran correctas:

$$\varphi_3 = \bigwedge_{h=1}^{k} \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=1\\ i\neq j}}^{n} \Big((x_{hi} \wedge x_{gj}) \rightarrow p_{ij} \vee p_{ji} \Big)$$

$$\varphi_3 = \bigwedge_{h=1}^{k} \bigwedge_{\substack{g=1\\ g\neq h}}^{k} \bigwedge_{i=1}^{n} \bigwedge_{\substack{j=1\\ i\neq j}}^{n} \Big((x_{hi} \wedge x_{gj}) \rightarrow p_{ij} \Big)$$

Definimos la fórmula φ como $\varphi = \psi \wedge \varphi_1 \wedge \varphi_2 \wedge \varphi_3$.

Para demostrar la correctitud de la fórmula, debemos demostrar dos direcciones:

($\Rightarrow$) PDQ. Si existe un núcleo con k personas para V y E, entonces la fórmula φ es satisfactible.

Supongamos que existe un núcleo con k personas para V y E y sea $C = \{c_1, ..., c_k\}$ dicho núcleo (de manera que cada $c_i \in V$). Haremos una demostración constructiva, es decir, construiremos una valuación σ a partir de C que satisface a φ. Definimos a σ sobre las variables proposicionales propuestas como sigue:

$$\sigma(p_{ij}) = \begin{cases} 1 & \text{si } (i,j) \in E \\ 0 & \text{en otro caso} \end{cases}$$

$$\sigma(x_{hi}) = \begin{cases} 1 & \text{si } c_h = i \\ 0 & \text{en otro caso} \end{cases}$$

Notemos entonces que:

- $\sigma(\psi) = 1$, pues, por construcción, para cada $(i,j) \in E$, $\sigma(p_{ij}) = 1$ y para cada $(i,j) \notin E$, $\sigma(\neg p_{ij}) = 0$, luego las conjunciones de todos los (i,j) serán también verdaderas. (Lo anterior también aplica cuando solo consideramos los $(i,j) \notin E$.)
- $\sigma(\varphi_1) = 1$, pues como a cada posición $1 \leq h \leq k$ del núcleo se le asigna exactamente una persona $i = c_h$, y por construcción sólo en ese caso $\sigma(x_{hi}) = 1$. Luego, para cada posición h se cumple que la disyunción de x_{hi} es verdadera y, cuando lo es, todos los demás $\sigma(x_{hj})$ toman valor 0, por lo que la conjunción en el consecuente es verdadera también, por lo que toda la fórmula se le asigna valor de verdad 1.
- $\sigma(\varphi_2) = 1$, pues para toda posición $h \in \{1, ..., k\}$ y persona $i \in \{1, ..., n\}$, si $c_h = i$ (es decir, la persona i es la h-ésima del núcleo), entonces $\sigma(x_{hi}) = 1$ y la persona i solo aparece una vez en el núcleo por lo que para todo $g \in \{1, ..., k\}$, se tiene que $\sigma(x_{gi}) = 0$ por construcción. Luego cada implicancia de la conjunción es cierta y por lo tanto la fórmula entera es verdadera.

- $\sigma(\varphi_3) = 1$, pues para todo par de personas en el núcleo c_h y c_g, se cumple que deben ser compatibles por definición de núcleo, es decir, si $c_h = i$ y $c_g = j$, con $i < j$, entonces $(i, j) \in E$. Luego, la implicancia en φ_3 expresa exactamente esto, pues si $c_h = i$ y $c_g = j$, entonces $\sigma(x_{hi}) = \sigma(x_{gj}) = 1$, y si $(i, j) \in E$, entonces $\sigma(p_{ij}) = 1$. Luego, cada implicancia en la conjunción es cierta y por lo tanto la fórmula entera es verdadera.

Finalmente, como $\sigma(\psi) = \sigma(\varphi_1) = \sigma(\varphi_2) = \sigma(\varphi_3) = 1$ y φ es la conjunción de estas fórmulas, se tiene que $\sigma(\varphi) = 1$ y por lo tanto que φ es satisfactible.

($\Leftarrow$) PDQ. Si la fórmula φ es satisfactible, entonces existe un núcleo para V y E con k personas.

Supongamos que φ es satisfactible, por lo que existe una valuación que la satisface, a la que llamaremos σ. Haremos acá igualmente una demostración constructiva, es decir, construiremos un núcleo C a partir de la valuación. Para lo anterior, definimos C como el conjunto de las personas i tales que para algún $h \in \{1, ..., k\}$ se tiene que $\sigma(x_{hi}) = 1$ (es decir, si $\sigma(x_{hi}) = 1$ entonces para algún h, entonces $i \in C$. Notemos entonces que $\sigma(\varphi) = 1$, por lo que $\sigma(\psi) = \sigma(\varphi_1) = \sigma(\varphi_2) = \sigma(\varphi_3) = 1$. Luego:

- Como $\sigma(\varphi_1) = 1$, hay exactamente k variables x_{hi} con asignación de verdad 1, por lo que C tiene k elementos (posiblemente repetidos). Como además $\sigma(\varphi_2) = 1$, cada uno de estos elementos es único, es decir, C tiene exactamente k elementos.
- Si tomamos dos personas del núcleo construido, i y j, por cómo construimos el núcleo sabemos que $\sigma(x_{hi}) = \sigma(x_{gi})$ para algunas posiciones h y g. Como $\sigma(\varphi_3) = 1$, se tiene que $\sigma(p_i j)$ debe ser 1. Luego, como $\sigma(\psi) = 1$, notamos que p_{ij} es verdadero solo cuando i y j son compatibles. Es decir, todo par de personas en el núcleo construido son compatibles.

En conclusión, el conjunto C construido es un conjunto de k personas compatibles entre sí, es decir un núcleo de tamaño k en V y E.

Concluimos que la fórmula es correcta.

Puntaje:

- 1 pto. por una fórmula que inicializa las variables p_{ij}
- 1 pto. por una fórmula que expresa que cada posición tiene exactamente una persona en el núcleo
- 1 pto. por una fórmula que expresa que en posiciones distintas hay personas distintas
- 1 pto. por una fórmula que expresa la compatibilidad entre personas del núcleo
- 1 pto. por mostrar que si el núcleo de tamaño k existe, la fórmula es satisfactible
- 1 pto. por mostrar que si la fórmula es satisfactible, el núcleo existe