P1

a) Los $\wedge$ de φ podemos usarlos para separar los elementos del conjunto, con esto:

$$\bigcup\{(l_i \vee l_i')\} \models \bigwedge(l_i \vee l_i')$$

Entonces $\Sigma = \bigcup\{(l_i \vee l_i')\}$

b) Primero que nada aclararemos notación:

P está en $(l_i \vee l_i') \iff l_i = P \vee l_i' = P$

$\mathbb{P} = \{P \mid \exists i \text{ t.q. } P \text{ está en } (l_i \vee l_i')\}$

Con esto, primero veremos que si existe algún $P \in \mathbb{P}$ tal que $\forall i$ $\neg P$ no está en $(l_i \vee l_i')$ esto quiere decir que cada vez que aparece P, este esta en forma positiva (no negado).

Como queremos determinar si φ es satisfacable, nos basta ponernos en un caso conveniente, por lo que podemos asignar que P será verdadero en nuestra valuación.

Con esto, $\forall i$ t.q. P está en $(l_i \vee l_i')$ como $\sigma(P) = 1$

$$\Rightarrow \sigma(l_i \vee l_i') = 1$$

entonces no es necesario considerar esas cláusulas, ya que en nuestra valuación ya son correctas.

y $\Sigma \models \Sigma \cup \{V\}$.

Claramente, lo mismo aplica para $\neg q$, tal que $\forall i$ q no está en $(l_i \vee l_i)$. Excepto por que en nuestra valuación $\sigma(q) = 0$.

Una vez dicho esto, podemos comenzar nuestro algoritmo eliminando todos los p que se pueda con este método.

Paso 1: mientras $\exists p$ ta $\forall i$ $\neg p$ no está en $(l_i \vee l_i')$
ó $\forall i$ p no está en $(l_i \vee l_i')$
Eliminamos todas las cláusulas en las que aparece p.

Una vez terminado el paso 1, tenemos un conjunto Σ' tal que q está en $\Sigma' \iff \neg q$ está en Σ'.

Paso 2: Ahora queremos determinar si Σ' es satisfacible, para esto usaremos el método de resolución:

Tomamos un P_K cualquiera que no haya sido eliminado en el paso 1.

Para cada pareja $(l_i \vee l_i')$ en la que esté P_K o $\neg P_K$, si el elemento que acompaña a P_K no es otro P_K vamos a usar la regla de la simplificación de la sgte. forma:

Sea $(P_K \vee l_i)$, tomamos todas las cláusulas tales que l_i esté en ellas y todas las que contengan a $\neg l_i$ usando la regla de la simplificación eliminamos todas las instancias de l_i:

$$\frac{(P_K \vee l_i) \quad (l_j \vee \neg l_i)}{P_K \vee l_j}$$

siempre $\neg l_i$ estará en alguna cláusula, si no se habría eliminado en el paso 1.

con este método eliminaremos variables l_i hasta que en toda cláusula en que esté P_K, estará sólo o con $\neg P_K$, lo mismo para $\neg P_K$.

Al final de este desarrollo hay 4 opciones:

1) Existen exclusivamente clausulas con P_K solo:
   $\Rightarrow \Sigma' \models P_K$, por lo tanto $\sigma(P_K)$ debe ser verdadero
2) Lo mismo para $\neg P_K$: es decir $\Sigma' \models \neg P_K$, $\sigma(P_K) = 0$
3) P_K queda en clausulas de la forma $(P_K \vee \neg P_K)$, las cuales son siempre verdaderas, esto quiere decir que pueden haber valuaciones que cumplan Σ' con P_K cierto y con P_K falso
4) Quedan clausulas de la forma $(P_K \vee P_K) \wedge (\neg P_K \vee \neg P_K)$:
   $\Sigma' \models \square$, ninguna valuación de P_K cumple con Σ'.

Si el resultado es 1, 2 ó 3, continúo con el siguiente P_K.
Si el resultado es 4 detengo el algoritmo, Σ no es satisfacible, ya que $\Sigma' \models \square$ y $\Sigma' \models \Sigma$!

Este algoritmo es más eficiente que probar todas las combinaciones, porque podemos acotar sus iteraciones por un número menor, esto no es relevante en esta parte del curso así que no se hará aquí.

P2

a) ¿→ es refleja? diremos que sí, porque si la página no tiene un link a sí misma tomaremos el botón de refrescar como un link.

¿→ es transitiva? No, pues Wikipedia → Google
Google → homerswebpage.com

pero Wikipedia no tiene link a la página web de homero :(.

¿→ es simétrica? Google → homerswebpag.com
¡NO! pero no en el sentido contrario

¿→ es antisimétrica? Bing → Google
¡No! Google → Bing

como no es transitiva, no es de orden ni de equivalencia.

b) $\forall w,e,b \in W \ (w \to e \wedge e \rightsquigarrow b) \Rightarrow (w \rightsquigarrow b)$
$\forall w \in W \ (w \rightsquigarrow w)$
Con estas dos proposiciones obtenemos lo pedido:

1) $(w \to e) \Rightarrow (w \rightsquigarrow e)$
si tomo $((w \to e) \wedge (w \rightsquigarrow w)) \Rightarrow (w \rightsquigarrow b)$

2) si $(w \to e) \wedge (e \rightsquigarrow b)$ clicamos el link de w a e y luego sigo el camino de e a b, es decir existe un camino $w \rightsquigarrow b$.
OjO: $\rightsquigarrow$ sigue sin ser de orden y sin ser simétrica ni antisimétrica.

c)

1) $\leftrightsquigarrow$ es refleja por construcción

2) $\leftrightsquigarrow$ es simétrica también se tiene por la construcción ya que $w \leftrightsquigarrow e \iff w \rightsquigarrow e \wedge e \rightsquigarrow w$

$\Rightarrow e \rightsquigarrow w \wedge w \rightsquigarrow e \iff e \leftrightsquigarrow w$

es decir, es simétrica

3) $\leftrightsquigarrow$ es transitiva porque:

$w \leftrightsquigarrow e \wedge e \leftrightsquigarrow b \iff w \rightsquigarrow e \wedge e \rightsquigarrow w \wedge e \rightsquigarrow b \wedge b \rightsquigarrow e$

Por transitividad de $\rightsquigarrow$ se tiene

$$\left.\begin{array}{l} w \rightsquigarrow e \wedge e \rightsquigarrow b \Rightarrow w \rightsquigarrow b \\ b \rightsquigarrow e \wedge e \rightsquigarrow w \Rightarrow b \rightsquigarrow w \end{array}\right\} w \leftrightsquigarrow e$$

$\therefore \leftrightsquigarrow$ es de orden

OjO: ¿Cualquier relación refleja y transitiva como $\rightsquigarrow$ puede generar una equivalencia?

d) Sea $W/\leftrightsquigarrow$ definimos $\Rightarrow$:

$$[P] \Rightarrow [q] \iff [P] \cap [q] \neq \emptyset$$

$$\exists p \in [P], q \in [q] \text{ t.a. } p \rightsquigarrow q$$

$\Rightarrow$ es de orden porque:

1) es refleja por construcción

2) es transitiva porque: $[P] \Rightarrow [q] \wedge [q] \Rightarrow [r]$

$\exists p, q, q', r : p \rightsquigarrow q \wedge q \rightsquigarrow r$

$q, q' \in [q] \Rightarrow q \leftrightsquigarrow q'$

$\Rightarrow p \rightsquigarrow q \wedge q \rightsquigarrow q' \wedge q' \rightsquigarrow r$

$\Rightarrow p \rightsquigarrow r \Rightarrow [P] \Rightarrow [r]$

3) es antisimétrica porque:

$$\underbrace{[P] \preceq_{\mathcal{B}} [q] \wedge [q] \preceq_{\mathcal{B}} [P]} \Rightarrow [q] = [P]$$

$\exists P, P', q, q' : P \rightsquigarrow q \wedge q' \rightsquigarrow P'$ pero $P, P' \in [P]$ y $q, q' \in [q]$

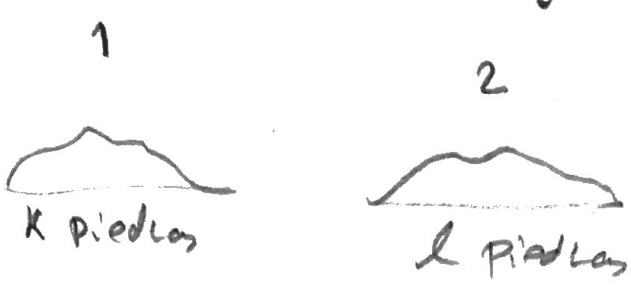
$\Rightarrow P \rightsquigarrow q \wedge q' \rightsquigarrow P' \wedge q \rightsquigarrow q' \wedge P' \rightsquigarrow q$, por transitividad:

$P \rightsquigarrow q \wedge q \rightsquigarrow P' \wedge P' \rightsquigarrow P \Rightarrow \underbrace{P \rightsquigarrow q \wedge q \rightsquigarrow P}$

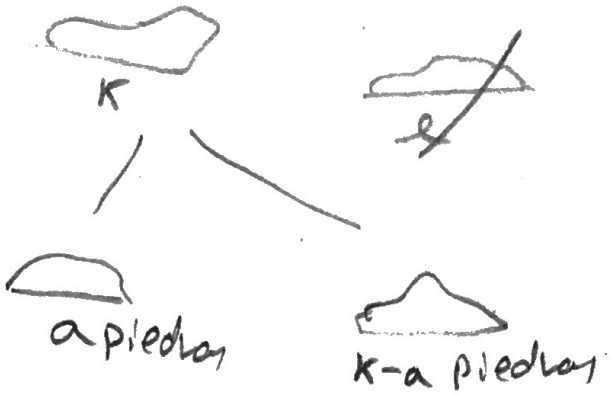
$P \circ\!\!\rightsquigarrow q$

$\Rightarrow [P] = [q]$ //

P3 En cada turno tengo 2 montones:



elimino uno y entrego 2 nuevamente



P. d. Q. Si parto con una cantidad de piedras par, gano:

Caso base: Si me entregan los montones 2 y K puedo eliminar el montón K y devolver los montones 1 y 1. Mi oponente toma una piedra, me devuelve la última, la tomo y gano.

Caso inductivo: Si me entregan 2 montones, uno de los cuales tiene cantidad par de piedras, puedo forzar a mi oponente a devolverme una cantidad par de piedras en un montón.

me entregan 2l y K, elimino K

Separo 2l en: l y l si l impar

l+1 y l-1 si l par

Ahora mi oponente recibe 2 opciones impares, luego de eliminar cualquiera debe separar un número impar de piedras en 2 montones, con lo que tiene 2 opciones:

a) devolver 0 y K, con lo que pierde

b) devolver a y K-a con lo que uno de ellos será par

Si se cumple a) gano y si se cumple b) sé que se eliminaron piedras, pero vuelvo al estado original, por lo que me estoy acercando al caso base.

∴ Si estoy en un estado par puedo volver a un estado par con menos piedras. El menor estado par es 2 y en ese caso gano.