ALGORITMIA Y COMPLEJIDAD

PUNCHED CARD PUZZLE - NPC

Salvador Carrillo Fuentes

UNIVERSIDAD DE MÁLAGA
E.T.S. INGENIERÍA INFORMÁTICA

PUNCH CARD PUZZLE

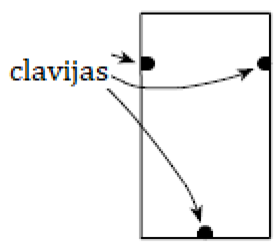
- Recuadro con **tres** clavijas
- Colección de $\boldsymbol{k}$ tarjetas $\langle \boldsymbol{t_1}, \ldots, \boldsymbol{t_k} \rangle$
- Cada tarjeta tiene **dos** columnas de círculos, que pueden perforarse (convirtiéndose en agujeros) o no
- Cada tarjeta es de color **granate** por una cara y **morado** por la otra. Por tanto, una tarjeta puede encajar en el recuadro de **dos** maneras posibles debido a sus tres muescas
- **Objetivo**: encajar todas las tarjetas en el recuadro, cubriendo completamente el fondo del mismo
- Se resuelve el puzle si todos los agujeros quedan tapados por alguna tarjeta que no tenga agujero en esa posición

PUNCH CARD PUZZLE

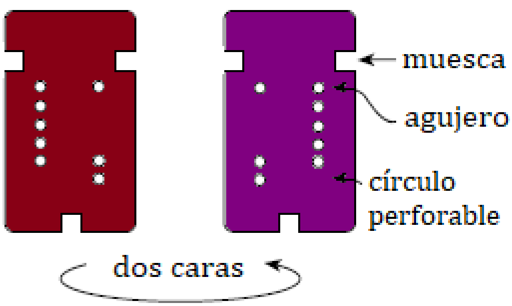
- Definición

$\boldsymbol{PUZZLE} = \{\langle \boldsymbol{t_1}, \ldots, \boldsymbol{t_k} \rangle \mid \text{cada } \boldsymbol{t_i} \text{ representa una tarjeta y esta colección es una solución}\}$

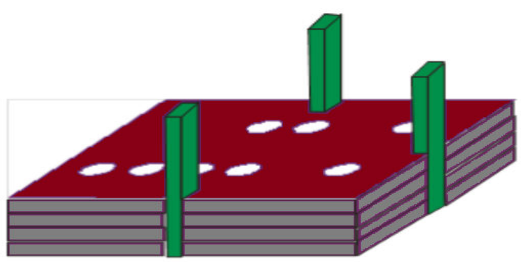
recuadro



tarjeta



- Ejemplo de instancia: colección de tarjetas con 6 filas en cada columna



$\boldsymbol{t_1}$ $\boldsymbol{t_2}$ $\boldsymbol{t_3}$ $\boldsymbol{t_4}$

PUNCH CARD PUZZLE ∈ NP

- Sea $\boldsymbol{M}$ una máquina de Turing no determinista que se ejecuta en tiempo polinomial para ***PUZZLE***:

- $\boldsymbol{M} =$ "Para la entrada $\langle \boldsymbol{t_1}, \ldots, \boldsymbol{t_k} \rangle$:
  1. De forma no determinista, seleccionar la orientación que tendrá cada tarjeta $\boldsymbol{t_i}$.
  2. Apilar todas las tarjetas (da igual el orden) y comprobar si el recuadro queda totalmente cubierto.
  3. Si el recuadro queda totalmente cubierto, *aceptar*; en otro caso, *rechazar*."

PUNCH CARD PUZZLE ∈ NP

- Sea V un verificador que trabaja en tiempo polinómico para $PUZZLE$:

- El certificado $\boldsymbol{c}$ es la colección de tarjetas orientadas de tal forma que se cubre todo el recuadro

- V= "Para la entrada $\langle\langle \boldsymbol{t_1}, \dots, \boldsymbol{t_k}\rangle, \boldsymbol{c}\rangle$:
    1. Comprobar si al encajar la colección orientada de tarjetas de $\boldsymbol{c}$ en el recuadro, este queda totalmente cubierto.
    2. Comprobar que $\boldsymbol{c}$ es una colección de tarjetas igual a $\langle \boldsymbol{t_1}, \dots, \boldsymbol{t_k}\rangle$.
    3. Si ambas comprobaciones son positivas, *aceptar*; en otro caso, *rechazar*."

- Apilar las tarjetas y comprobar si el recuadro queda totalmente cubierto se consigue en $\boldsymbol{\mathcal{O}(k)}$

3SAT

- Caso especial de $\boldsymbol{SAT}$ en el que todas las fórmulas están en una forma especial
- Un **literal** es una variable Booleana o su negación: $\boldsymbol{x}, \overline{\boldsymbol{x}}$
- Una **cláusula** es la conexión de varios literales con $\lor'$ $\boldsymbol{s}$, como en:

$$(\boldsymbol{x_1} \lor \overline{\boldsymbol{x_2}} \lor \overline{\boldsymbol{x_3}} \lor \boldsymbol{x_4})$$

- Una fórmula Booleana está en forma normal conjuntiva, llamada ***fórmula-fnc***, si está formada por varias cláusulas conectadas por $\land$ ‘$\boldsymbol{s}$:

$$(\boldsymbol{x_1} \lor \overline{\boldsymbol{x_2}} \lor \overline{\boldsymbol{x_3}} \lor \boldsymbol{x_4}) \land (\boldsymbol{x_3} \lor \overline{\boldsymbol{x_5}} \lor \boldsymbol{x_6}) \land (\boldsymbol{x_3} \lor \overline{\boldsymbol{x_6}})$$

- Tenemos una ***fórmula-3fnc*** si todas las cláusulas tiene **tres** literales:

$$(\boldsymbol{x_1} \lor \overline{\boldsymbol{x_2}} \lor \overline{\boldsymbol{x_3}}) \land (\boldsymbol{x_3} \lor \overline{\boldsymbol{x_5}} \lor \boldsymbol{x_6}) \land (\boldsymbol{x_3} \lor \overline{\boldsymbol{x_6}} \lor \boldsymbol{x_4})$$

- Sea $\boldsymbol{3SAT} = \{\langle \boldsymbol{\phi} \rangle \mid \boldsymbol{\phi} \text{ es una } \boldsymbol{f}\text{ó}\boldsymbol{rmula} - \boldsymbol{3fnc} \text{ satisfacible}\}$
  Cada cláusula debe tener al menos un literal evaluado a ***true***
- Tomamos como resultado previo que ***3SAT*** $\in$ **NPC**

$\mathbf{PUZZLE} \in \mathbf{NP} - \mathbf{completo}$

- Debemos reducir ***3SAT*** a ***PUZZLE***

$$(\boldsymbol{3SAT} \leq_{\boldsymbol{P}} \boldsymbol{PUZZLE})$$

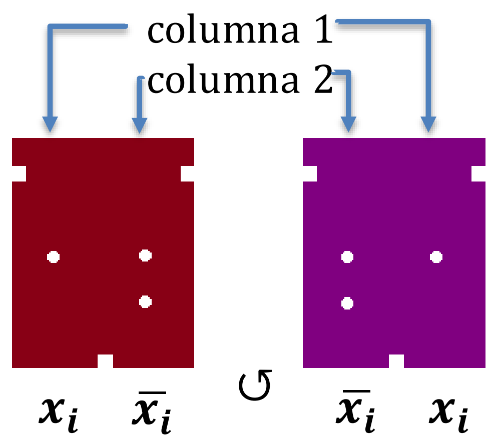
- Hay que encontrar una función computable en tiempo polinómico $\boldsymbol{f}: \boldsymbol{\Sigma}^* \longrightarrow \boldsymbol{\Sigma}^*$, tal que, para cada ***w***,

$$\boldsymbol{w} \in \boldsymbol{3SAT} \Leftrightarrow \boldsymbol{f}(\boldsymbol{w}) \in \boldsymbol{PUZZLE}$$

- Tenemos que encontrar una estructura en ***PUZZLE*** que simule las variables y cláusulas de la $\boldsymbol{f}$ó$\boldsymbol{rmula} - \boldsymbol{3fnc}$

PUZZLE $\in$ NP – completo

- Dada una fórmula Booleana $\boldsymbol{\phi}$, en $\boldsymbol{3fnc}$, podemos transformarla en una colección de tarjetas
- Sean $\boldsymbol{x_1, x_2, \ldots, x_m}$ las variables de $\boldsymbol{\phi}$ y $\boldsymbol{c_1, c_2, \ldots, c_l}$ sus cláusulas
- Decimos que $\boldsymbol{z} \in \boldsymbol{c_j}$ si $\boldsymbol{z}$ es uno de los literales de la cláusula $\boldsymbol{c_j}$
- Cada tarjeta tiene una **columna 1** y una **columna 2**, con $\boldsymbol{l}$ filas de posibles agujeros (pueden ser perforados o no) en cada columna, enumerados desde 1 hasta $\boldsymbol{l}$
- Al encajar una tarjeta en el recuadro, podemos hacerlo poniendo el lado **granate hacia arriba** o el lado **morado arriba**.
- $\begin{cases} \textit{granate hacia arriba} \begin{cases} \textit{columna } 1\text{: } \textit{izquierda} \\ \textit{columna } 2\text{: } \textit{derecha} \end{cases} \\ \textit{morado hacia arriba} \begin{cases} \textit{columna } 1\text{: } \textit{derecha} \\ \textit{columna } 2\text{: } \textit{izquierda} \end{cases} \end{cases}$



$\mathbf{PUZZLE} \in \mathbf{NP-completo}$

- Sea $\boldsymbol{F}$ un algoritmo que computa la reducción $\boldsymbol{3SAT \leq_P PUZZLE}$:
- $\boldsymbol{F} =$ "Para la entrada $\boldsymbol{\phi}$, siendo una $\boldsymbol{f}$ó$\boldsymbol{rmula - 3fnc}$ ":

  1. Crea una tarjeta $\boldsymbol{t_i}$ para cada variable $\boldsymbol{x_i}$ de la siguiente manera:
     - En la **columna 1** perfora todas la posiciones menos aquellas posiciones en la fila $\boldsymbol{j}$ tal que $\boldsymbol{x_i} \in \boldsymbol{c_j}$ (cláusula $\boldsymbol{j}$).
     - En la **columna 2** perfora todas la posiciones menos aquellas posiciones en la fila $\boldsymbol{j'}$ tal que $\boldsymbol{\bar{x}_i} \in \boldsymbol{c_{j'}}$.

     (Cada fila corresponde con una cláusula de $\boldsymbol{\phi}$)

  2. Crea una tarjeta **extra** con todas las posiciones de la **columna 1** perforadas y ninguna posición de la **columna 2** perforada.

  3. Ofrece como salida la descripción de las tarjetas creadas."

PUZZLE $\in$ NP – completo

- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:

  - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

De esta forma, todos los agujeros de la **columna 1** quedarán cubiertos, porque la cláusula asociada tiene algún literal asignado a ***true***, y todos los agujeros de la **columna 2** quedarán cubiertos por la tarjeta **extra**.

PUZZLE $\in$ NP – completo

- (←) Dada una solución a *PUZZLE*, podemos construir una asignación que satisface a $\boldsymbol{\phi}$:

  - Damos la vuelta a la baraja de tarjetas para que la tarjeta **extra** cubra la **columna 2** con la cara granate hacia arriba (si fuera necesario).
  - Asignamos las variables según la orientación de las tarjetas, ***true*** cuando esté **hacia arriba la cara granate** y ***false*** cuando esté **hacia abajo**.

Como todos los agujeros de la **columna 1** están cubiertos, cada cláusula debe contener al menos un literal asignado a ***true***, lo que satisface a $\boldsymbol{\phi}$.

EJEMPLO

- Vamos a transformar una *fórmula-3fnc* en una baraja de tarjetas:

- Sea
$$\boldsymbol{\phi} = (\overline{\boldsymbol{x}} \vee \boldsymbol{y} \vee \overline{\boldsymbol{y}}) \wedge (\overline{\boldsymbol{x}} \vee \boldsymbol{z} \vee \overline{\boldsymbol{w}}) \wedge (\boldsymbol{y} \vee \boldsymbol{z} \vee \boldsymbol{w})$$

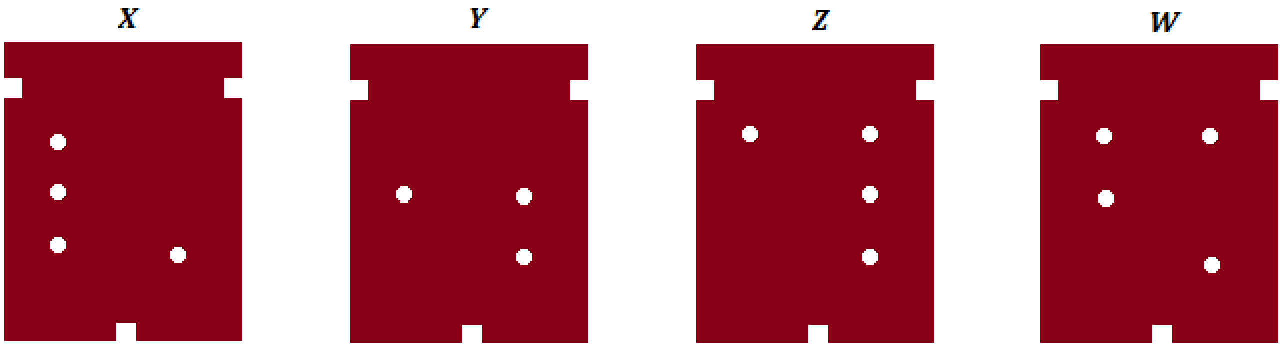
4 variables → **4** tarjetas

3 cláusulas → **3** filas de posibles agujeros

EJEMPLO

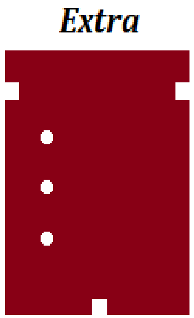
$$\boldsymbol{\phi} = (\bar{\boldsymbol{x}} \vee \boldsymbol{y} \vee \bar{\boldsymbol{y}}) \wedge (\bar{\boldsymbol{x}} \vee \boldsymbol{z} \vee \bar{\boldsymbol{w}}) \wedge (\boldsymbol{y} \vee \boldsymbol{z} \vee \boldsymbol{w})$$

1. Crea una tarjeta $\boldsymbol{t_i}$ para cada variable $\boldsymbol{x_i}$ de la siguiente manera:
   - En la **columna 1** perfora todas la posiciones menos aquellas posiciones en la fila $\boldsymbol{j}$ tal que $\boldsymbol{x_i} \in \boldsymbol{c_j}$ (cláusula $\boldsymbol{j}$).
   - En la **columna 2** perfora todas la posiciones menos aquellas posiciones en la fila $\boldsymbol{j'}$ tal que $\bar{\boldsymbol{x}}_i \in \boldsymbol{c_{j'}}$.

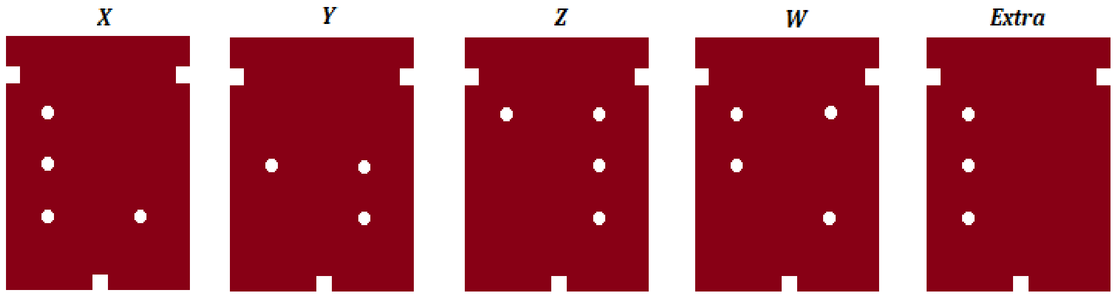


EJEMPLO

2. Crea una tarjeta **extra** con todas las posiciones de la **columna 1** perforadas y ninguna posición de la **columna 2** perforada.



3. Ofrece como salida la descripción de las tarjetas creadas:



EJEMPLO

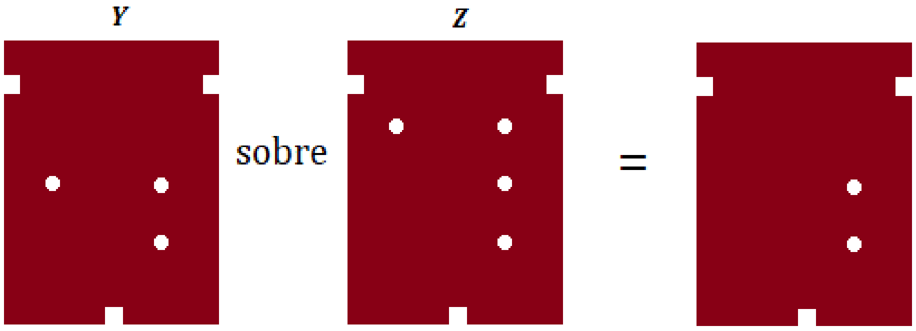
$$\boldsymbol{\phi} = (\overline{\boldsymbol{x}} \vee \boldsymbol{y} \vee \overline{\boldsymbol{y}}) \wedge (\overline{\boldsymbol{x}} \vee \boldsymbol{z} \vee \overline{\boldsymbol{w}}) \wedge (\boldsymbol{y} \vee \boldsymbol{z} \vee \boldsymbol{w})$$

Dada una asignación satisfacible:

$$\boldsymbol{y} = \boldsymbol{z} = \boldsymbol{true}$$

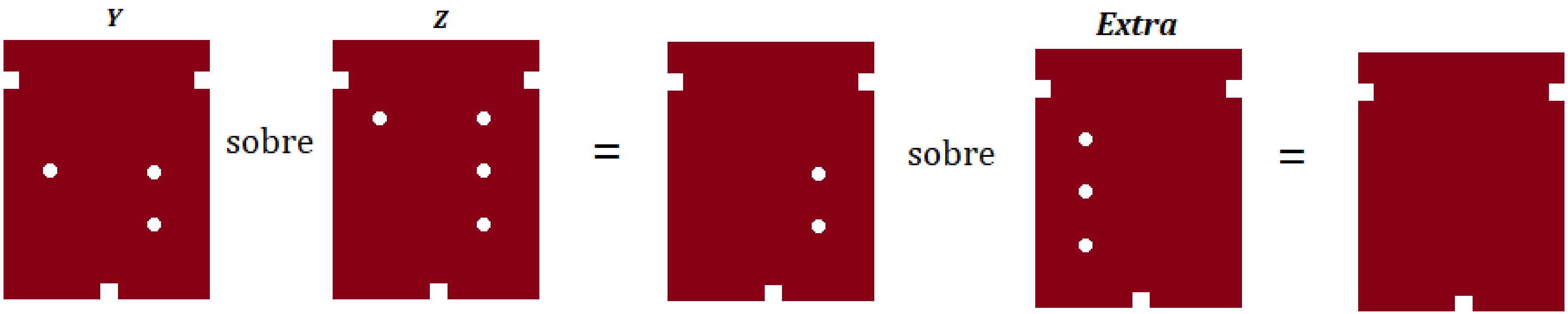
$$\boldsymbol{x}, \boldsymbol{w} = \text{cualquiera}$$

- Traducido a las tarjetas, esto significa que podemos tapar la columna 1 usando solo las tarjetas que representan a las variables $\boldsymbol{y}, \boldsymbol{z}$ con la cara granate hacia arriba:



EJEMPLO

- Ahora podríamos añadir la tarjeta extra y quedarían todos los huecos cubiertos:



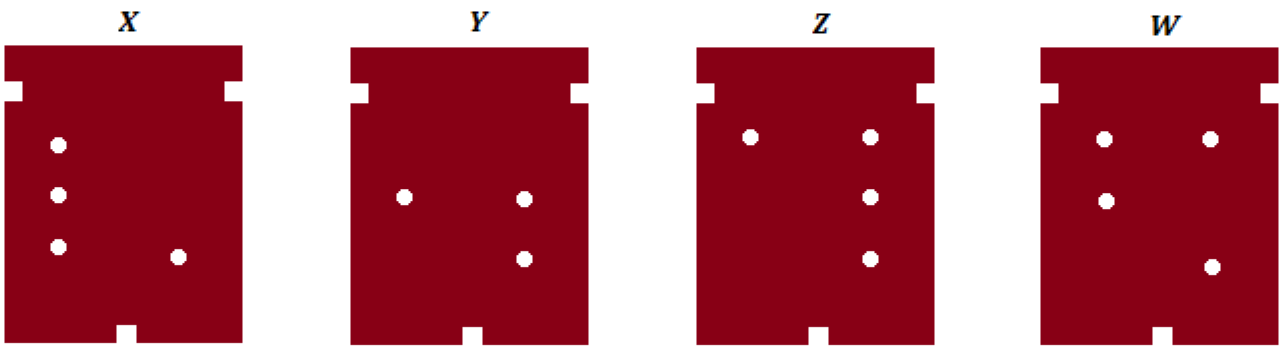
- Pero no nos hace falta la tarjeta extra porque podemos usar las tarjetas ***x*, *y*, *z***, y la ***w*** volteada (cara morada hacia arriba)

EJEMPLO

- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:

  - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

Asignación: $\boldsymbol{x = y = z = true}$; $\boldsymbol{w = false}$

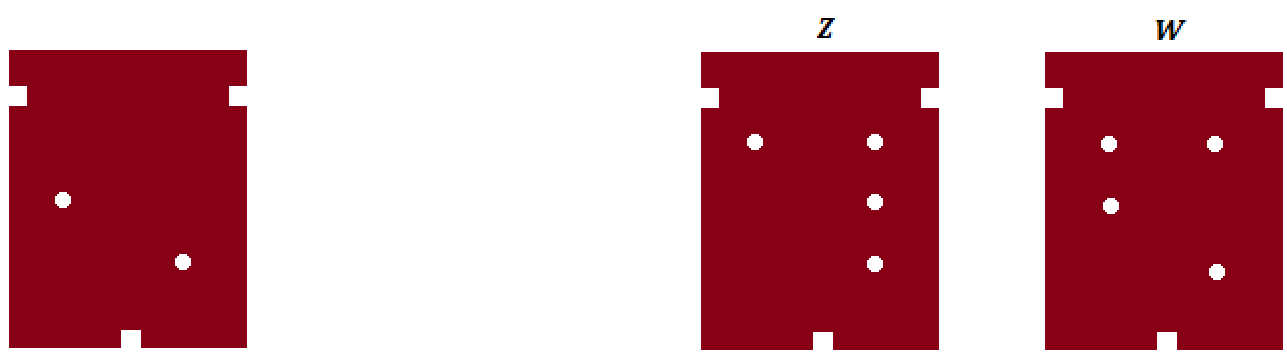


EJEMPLO

- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:

  - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

Asignación: $\boldsymbol{x = y = z = true};\ \boldsymbol{w = false}$

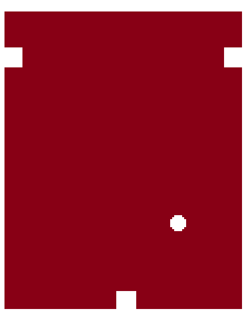


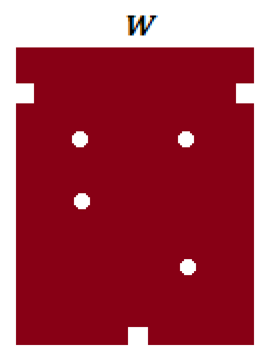
EJEMPLO

- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:

    - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

Asignación: $\boldsymbol{x = y = z = true}$; $\boldsymbol{w = false}$

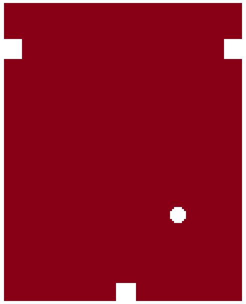


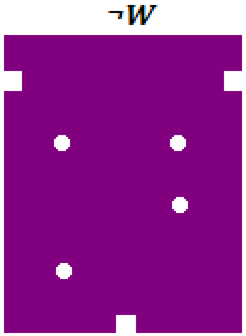


EJEMPLO

- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:
  - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

Asignación: $\boldsymbol{x = y = z = true}$; $\boldsymbol{w = false}$





EJEMPLO

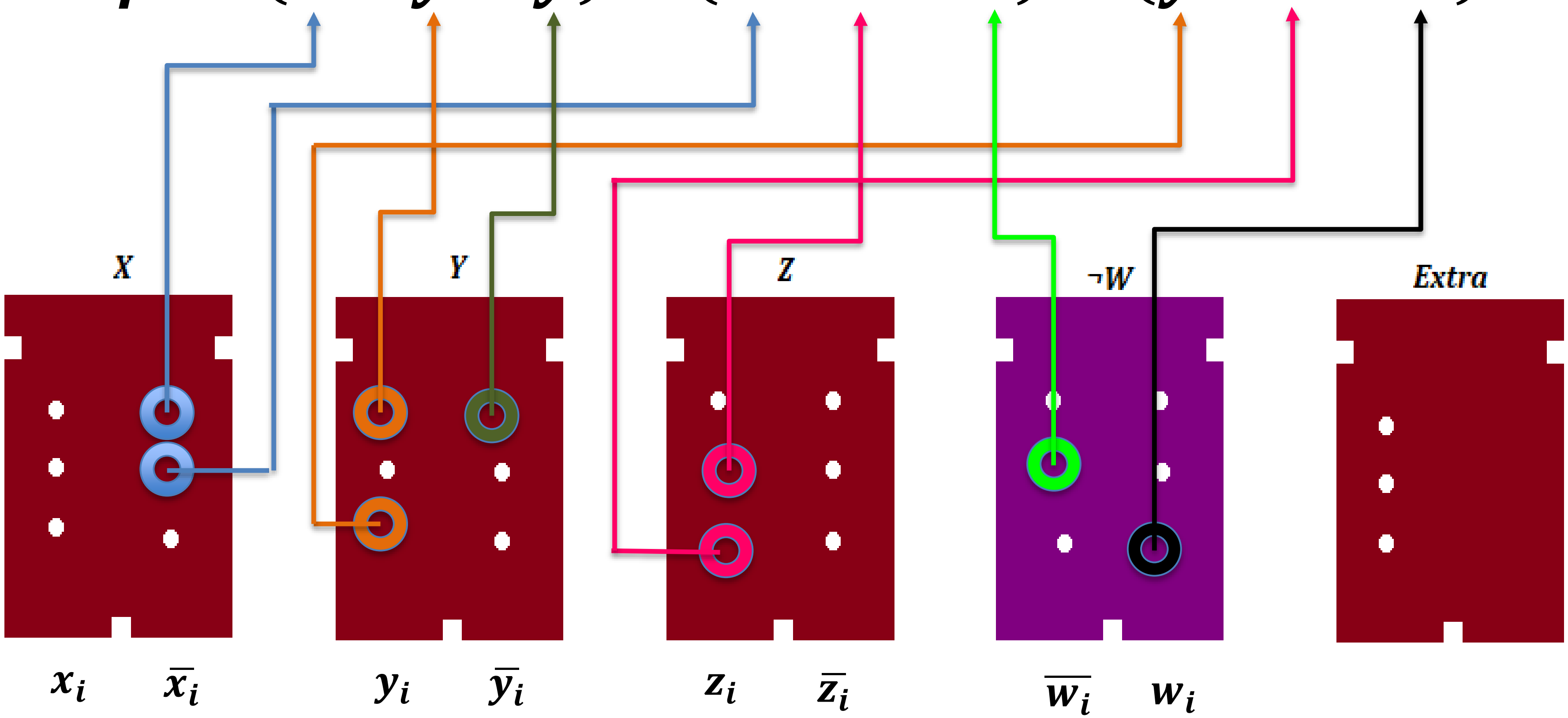
- (→) Dada una asignación que satisface a $\boldsymbol{\phi}$, podemos construir una solución a *PUZZLE*:

    - Para cada $\boldsymbol{x_i}$ asignado a ***true***, ponemos la tarjeta con la **cara granate hacia arriba**, poniéndola **hacia abajo** si la asignación es ***false***, y ponemos la tarjeta **extra** con la **cara granate hacia arriba**.

Asignación: $\boldsymbol{x = y = z = true};\ \boldsymbol{w = false}$

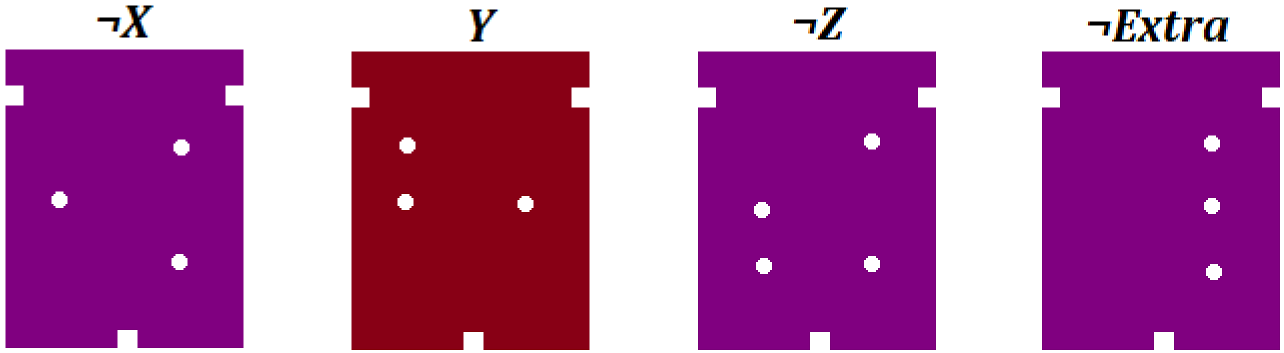
EJEMPLO

$$\phi = (\overline{x} \vee y \vee \overline{y}) \wedge (\overline{x} \vee z \vee \overline{w}) \wedge (y \vee z \vee w)$$



EJEMPLO

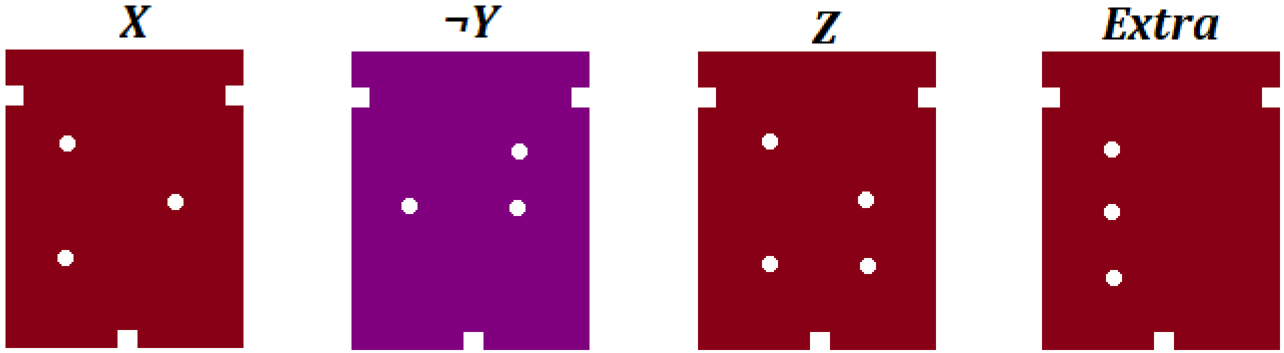
- (←) Dada una solución a *PUZZLE*, podemos construir una asignación que satisface a $\boldsymbol{\phi}$:
  - Damos la vuelta a la baraja de tarjetas para que la tarjeta **extra** cubra la **columna 2** con la cara granate hacia arriba (si fuera necesario).
  - Asignamos las variables según la orientación de las tarjetas, ***true*** cuando esté **hacia arriba** y ***false*** cuando esté **hacia abajo**.



$$\boldsymbol{\phi} = (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (x \vee z \vee z) \wedge (\bar{x} \vee y \vee \bar{y})$$

EJEMPLO

- (←) Dada una solución a *PUZZLE*, podemos construir una asignación que satisface a $\boldsymbol{\phi}$:
  - Damos la vuelta a la baraja de tarjetas para que la tarjeta **extra** cubra la **columna 2** con la cara granate hacia arriba (si fuera necesario).
  - Asignamos las variables según la orientación de las tarjetas, ***true*** cuando esté **hacia arriba** y ***false*** cuando esté **hacia abajo**.



Asignación:
$\boldsymbol{x, z = true}$
$\boldsymbol{y = false}$

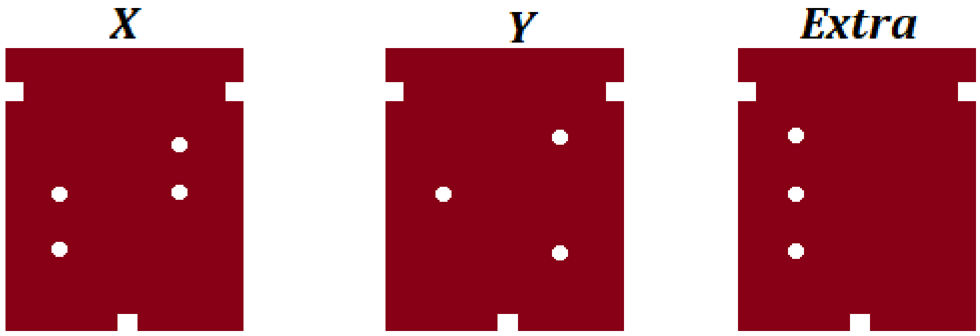
$$\boldsymbol{\phi} = (\overline{\boldsymbol{x}} \vee \overline{\boldsymbol{y}} \vee \overline{\boldsymbol{z}}) \wedge (\boldsymbol{x} \vee \boldsymbol{z} \vee \boldsymbol{z}) \wedge (\overline{\boldsymbol{x}} \vee \boldsymbol{y} \vee \overline{\boldsymbol{y}})$$

EJEMPLO

- El ejemplo **insatisfacible** más simple en el que no se repiten variables en las cláusulas requiere tres variables, 2^3 cláusulas.

- Ejemplo insatisfacible (repitiendo literales):

$$\boldsymbol{\phi} = (\boldsymbol{x} \vee \boldsymbol{y} \vee \boldsymbol{y}) \wedge (\overline{\boldsymbol{y}} \vee \overline{\boldsymbol{y}} \vee \overline{\boldsymbol{y}}) \wedge (\overline{\boldsymbol{x}} \vee \boldsymbol{y} \vee \boldsymbol{y})$$

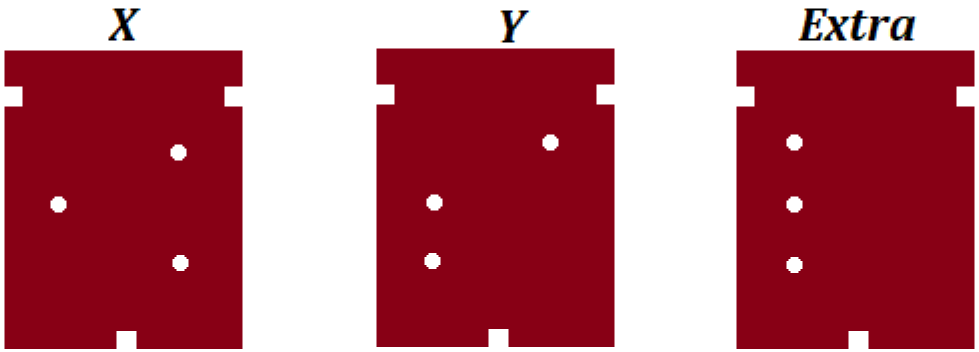


- Si $\boldsymbol{\phi}$ es insatisfacible, ninguna columna se cubre. Da igual que gire cualquier de las tarjetas. Sólo si una columna se puede cubrir, $\boldsymbol{\phi}$ será satisfacible.

EJEMPLO

- Ejemplo en el que es necesario usar la tarjeta extra.

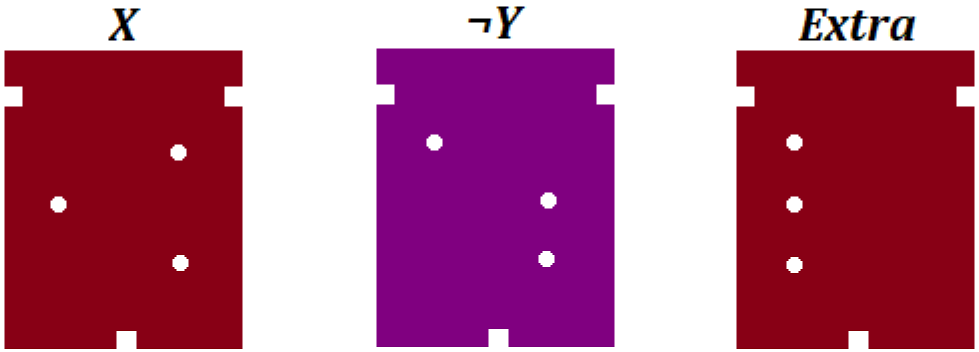
$$\boldsymbol{\phi} = (\boldsymbol{x} \vee \boldsymbol{y} \vee \boldsymbol{y}) \wedge (\overline{\boldsymbol{x}} \vee \overline{\boldsymbol{y}} \vee \overline{\boldsymbol{y}}) \wedge (\boldsymbol{x} \vee \overline{\boldsymbol{y}} \vee \overline{\boldsymbol{y}})$$



EJEMPLO

- Ejemplo en el que es necesario usar la tarjeta extra.

$$\phi = (x \vee y \vee y) \wedge (\overline{x} \vee \overline{y} \vee \overline{y}) \wedge (x \vee \overline{y} \vee \overline{y})$$



Fuentes

- http://magni.strumpur.net/Hi/rrr/skil11.pdf