A lo largo de este documento, nos referiremos a dos fórmulas:

- $\varphi_1 \equiv (x_1 + x_2) \cdot (\overline{x}_1 + x_2) \cdot (x_1 + \overline{x}_2)$
- $\varphi_2 \equiv (x_1 + x_2) \cdot (\overline{x}_1 + x_2) \cdot (x_1 + \overline{x}_2) \cdot (\overline{x}_1 + \overline{x}_2)$

Es fácil darse cuenta que φ_1 es una fórmula satisfactible (de hecho, sólo tiene una valoración de verdad que la hace verdadera, $\sigma(x_1) = \sigma(x_2) = 0$), mientras que φ_2 **no** es una fórmula satisfactible, dado que no existe ninguna valoración de verdad que haga verdadera la fórmula.

En el tema 5: Problemas **NP**-completos hemos visto la demostración de la **NP**-completitud del problema `SAT`. Para ello, primero se demuestra que dicho problema pertenece a la clase **NP**. Para ello, se define un algoritmo no determinista $\mathcal{A}$ que resuelve el problema `SAT`. Podemos entender este tipo de algoritmos desde dos puntos de vista, equivalentes entre sí:

1. La primera, es pensar en que la computación se va a bifurcar cada vez que se use la función `elegir`, creando así un árbol de computaciones. Cada rama devolverá un resultado y, en el caso en el que alguna de las ramas devuelva `sí`, entonces el algoritmo **acepta** el dato de entrada. Esto se puede observar en las Figuras 1 y 2.

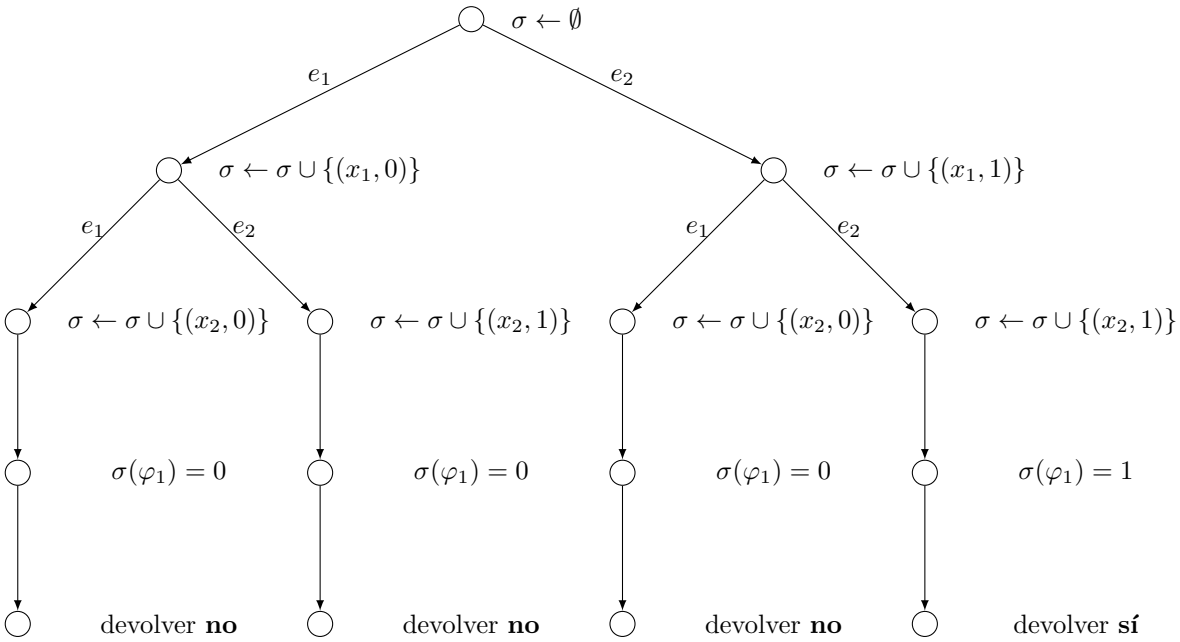

Figura 1: Árbol de computación del algoritmo $\mathcal{A}$ con dato de entrada φ_1

2. La segunda visión es pensar que un algoritmo no determinista es un algoritmo muy inteligente que, si existe en este caso una valoración de verdad

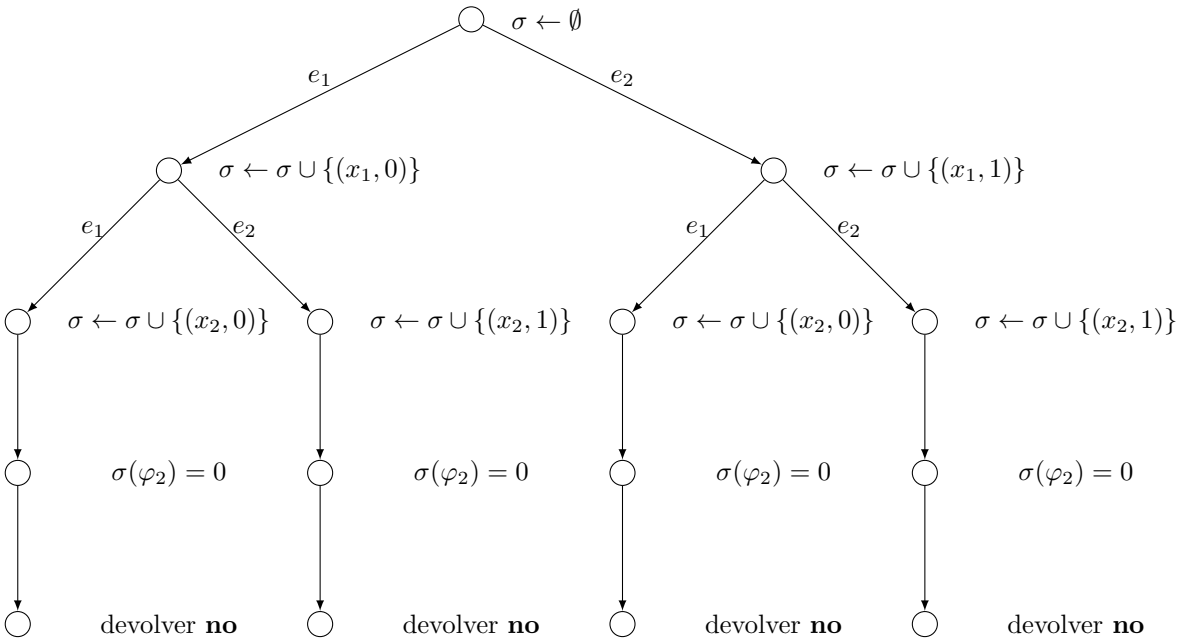

Figura 2: Árbol de computación del algoritmo $\mathcal{A}$ con dato de entrada φ_2

que haga verdadera la fórmula proposicional, la función `elegir` va a ir escogiendo la opción acertada; pero es importante esto, va a elegir una valoración que haga verdadera la fórmula **si existe**, en caso contrario, va a elegir una cualquiera. Con esto, es fácil darse cuenta que, si nos devuelve **sí** es porque ha encontrado una valoración de verdad que hace verdadera la fórmula; si nos devuelve **no**, podemos asegurar que no existe ninguna valoración de verdad que haga verdadera la fórmula, en caso contrario el algoritmo habría seleccionado la correcta y habría devuelto **sí**. Esto se puede observar en las Figuras 3 y 4.

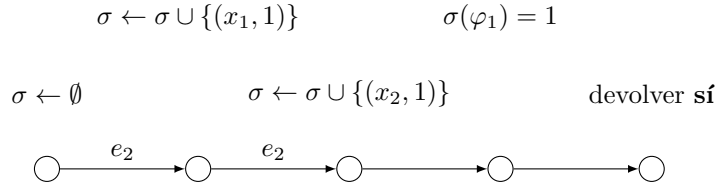

Figura 3: Computación «inteligente» del algoritmo $\mathcal{A}$ con dato de entrada φ_1

La segunda parte de la demostración es verificar que `SAT`es un problema **NP**-completo; es decir, verificar que `SAT`es un problema **más difícil** que todos los problemas de la clase **NP**. Podríamos usar el teorema de generación de

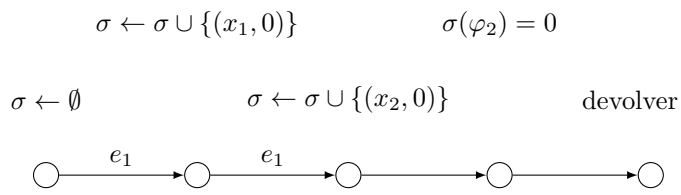

Figura 4: Computación «inteligente» del algoritmo $\mathcal{A}$ con dato de entrada φ_2

problemas **NP**-completos y, de una manera medianamente sencilla podríamos obtener el resultado... ¡Pero en 1971 S. Cook no tenía ningún otro problema **NP**-completo que reducir al problema `SAT` ! Por lo tanto, hay que buscar otra vía. Y esa otra vía es la siguiente:

Tenemos que demostrar que todos los problemas de la clase **NP**son más fáciles que el problema `SAT`(o, dicho de otro modo, que el problema `SAT`es más difícil que todos los problemas de la clase **NP**). Es decir, que

$$\forall X \in \mathbf{NP} : X \leq^p \texttt{SAT}$$

. Dado que $X \in \mathbf{NP}$, existirá una MTND, M, que trabaja en tiempo polinomial y resuelve X; es decir, existirá $k \geq 1$ tal que:

- Para cada $u \in \Sigma_X^*$, se tiene que $\theta_X(u) = 1$ **sii** existe alguna computación de $M(u)$ que devuelve **sí** en, a lo sumo, $|u|^k$ pasos.

Se imponen una serie de restricciones que serán útiles para la demostración:

1. $\Gamma = \Sigma_X$ y $F = \{q_y, q_n\}$.
2. M tiene una sóla cinta infinita con primera casilla.
3. La función de transición δ es total.
4. Aceptar $\equiv q_y$; Parar y no aceptar $\equiv q_n$.
5. En la primera casilla de la configuración inicial aparecerá el primer símbolo del dato de entrada.
6. En la configuración de aceptación, el cabezal analiza/inspecciona la primera casilla y todas las casillas están en blanco.

Vamos a considerar las siguientes fórmulas:

- $A(i, q) \equiv$ en el instante i, el estado de M es q.
- $B(i, j) \equiv$ en el instante i, el cabezal analiza/inspecciona la casilla j.
- $C(i, j, s) \equiv$ en el instante i, la casilla j contiene el símbolo s.

Supongamos que tenemos una máquina de Turing no determinista (vamos a usar una determinista que es un caso particular de no determinista por simplificar la construcción de la fórmula) M_1 definida como sigue:

$$M_1 = (Q, \Gamma, q_0, F, B, \delta)$$

con los siguientes elementos:

- $Q = \{q_0, q_y, q_n\}$;
- $\Gamma = \{0, 1\}$;
- $F = \{q_y, q_n\}$;
- $\delta = \{((q_0, 0), (q_n, 0, 0)), ((q_0, 1), (q_y, B, 0)), ((q_0, B), (q_0, B, 0))\}$ (aunque la última transición nunca se tenga que usar, es necesario ya que de lo contrario δ no sería una función **total**).

Esta máquina de Turing va a resolver el problema de la unidad $X = (\Sigma_X, \theta_X)$, definido como sigue:

- $\Sigma_X = \{0, 1\}$;
- $\theta_X(u) = u, u \in \Sigma_X$.

La máquina, obviamente, parará en exactamente un paso de computación. Entonces, vamos a ver que la fórmula generada φ_1 que simula todas las computacionales de M_1 sobre $u = u_1 \dots u_n$ es satisfactible si y solo si la máquina de Turing no determinista tiene, al menos, una computación de aceptación. Vayamos ecuación por ecuación:

1. **Configuración inicial:** dada la construcción de la fórmula, la subfórmula (1) siempre va a ser satisfactible, dado que el en la configuración $\mathcal{C}_0$, la máquina se encuentra en el estado q_0 ($A(0, q_0)$), el cabezal se encuentra analizando la primera casilla ($B(0, 1)$), las primeras n casillas están ocupadas por los símbolos $u_1 \dots u_n$ ($\bigwedge_{1 \leq j \leq |u|} C(0, j, u_j)$) y el resto de casillas tendrá símbolos en blanco ($\bigwedge_{|u|+1 \leq j \leq |u|^k+1} C(0, j, B)$).

2. **Configuración de aceptación:** recordemos que la restricción 6 nos pide que la máquina de Turing no determinista M_1, en la configuración de aceptación ($A(|u|^k, q_y)$) tiene que estar analizando la primera casilla ($B(|u|^k, 1)$) y todas las casillas tienen que estar en blanco ($\bigwedge_{1 \leq j \leq |u|^k} C(0, j, B)$). Precisamente, $A(|u|^k, q_y)$ será cierto si y solo si una de las computaciones que simula es de aceptación.

3. M_1 **está en un solo estado:** esto siempre será cierto debido al funcionamiento de una máquina de Turing: en cada paso de computación, la función de transición δ se aplica a la configuración $\mathcal{C}_t$ para dar paso a $\mathcal{C}_{t+1}$, de tal manera que si el estado en la configuración $\mathcal{C}_t$ es q y está analizando el símbolo s, en la configuración siguiente mediante la aplicación de $\delta(q,s) = (q',s',r), m \in \{0,+1,-1\}$, en la configuración $\mathcal{C}_{t+1}$ la máquina escribirá el símbolo s' en la casilla que estaba analizando, pasará al estado q' y se moverá de acuerdo a r. Entonces, para cada configuración $\mathcal{C}_i, 0 \leq i \leq |u|^k$, se cumple que está en un solo estado $(\bigvee_{q \in Q} A(i,q))$ y no estará en dos estados a la vez $(\bigwedge_{p,q \in Q, p \neq q} [(\neg A(i,p)) \vee (\neg A(i,q))])$.

4. **Cada casilla tiene un solo símbolo:** esto siempre será cierto debido al funcionamiento de una máquina de Turing: en cada paso de computación, la función de transición δ se aplica a la configuración $\mathcal{C}_t$ para dar paso a $\mathcal{C}_{t+1}$, de tal manera que si el estado en la configuración $\mathcal{C}_t$ es q y está analizando el símbolo s, en la configuración siguiente mediante la aplicación de $\delta(q,s) = (q',s',r), m \in \{0,+1,-1\}$, en la configuración $\mathcal{C}_{t+1}$ la máquina escribirá el símbolo s' en la casilla que estaba analizando, pasará al estado q' y se moverá de acuerdo a r. Entonces, para cada configuración $\mathcal{C}_i, 0 \leq i \leq |u|^k$, cada casilla $j, 1 \leq j \leq |u|^k + 1$ tiene un solo símbolo $(\bigvee_{s \in \Sigma_X} B(i,j))$ y no tendrá dos símbolos a la vez $(\bigwedge_{s,t \in \Sigma_X, s \neq t} [(\neg C(i,j,s)) \vee (\neg C(i,j,t))])$.

5. **El cabezal analiza una sola casilla:** esto siempre será cierto debido al funcionamiento de una máquina de Turing: en cada paso de computación, la función de transición δ se aplica a la configuración $\mathcal{C}_t$ para dar paso a $\mathcal{C}_{t+1}$, de tal manera que si el estado en la configuración $\mathcal{C}_t$ es q y está analizando el símbolo s, en la configuración siguiente mediante la aplicación de $\delta(q,s) = (q',s',r), m \in \{0,+1,-1\}$, en la configuración $\mathcal{C}_{t+1}$ la máquina escribirá el símbolo s' en la casilla que estaba analizando, pasará al estado q' y se moverá de acuerdo a r. Entonces, para cada configuración $\mathcal{C}_i, 0 \leq i \leq |u|^k$, se cumple que el cabezal está analizando alguna casilla $(\bigvee_{1 \leq j \leq |u|^k + 1} B(i,j))$ y no estará analizando dos casillas a la vez $(\bigwedge_{1 \leq j,j' \leq |u|^k + 1, j \neq j'} [(\neg B(i,j)) \vee (\neg B(i,j'))])$.

6. **La función de transición pasa de $\mathcal{C}_i$ a $\mathcal{C}_{i+1}$:** esto siempre será cierto, debido al funcionamiento de una máquina de Turing: si la máquina en el instante i está en el estado q $(A(i,q))$, el cabezal está leyendo la casilla j $(B(i,j))$ y esa casilla tiene el símbolo s $(C(i,j,s))$ entonces ocurrirá alguna de las posibles transiciones (recordemos que, dado que M_1 es una máquina de Turing no determinista, $\delta(q,s)$ puede dar lugar a distintos comportamiento); es decir, en la configuración siguiente $i+1$, la máquina estará en algún estado q' $(A(i+1,q'))$, estará leyendo una casilla

$j + r$ ($B(i + 1, j + r)$) y el símbolo de la casilla j habrá cambiado al s' correspondiente ($C(i + 1, j, s')$).

7. **Si una casilla no se analiza conserva su símbolo:** esto siempre será cierto debido al funcionamiento de una máquina de Turing: si en una configuración ($C(i, j, s)$) una casilla no está siendo analizada ($\neg B(i, j)$), el símbolo de esa casilla no puede cambiar en el paso de computación correspondiente ($C(i + 1, j, s)$). En la diapositiva 20 del Tema 5 aparecía la subfórmula $C(i, j, s) \wedge \neg B(i, j) \wedge C(i+1, j, s)$, que es errónea. La fórmula correcta es $C(i, j, s) \wedge \neg B(i, j) \rightarrow C(i + 1, j, s)$

Teniendo presente que la fórmula φ_1 correspondiente es una FNC de una conjunción de los anteriores, y teniendo presente que las subfórmulas (1), (3), (4), (5), (6) y (7) siempre se cumplen, nos queda por revisar la subfórmula (2). Pero precisamente esta subfórmula es la que se refiere a la condición de aceptación; es decir, sólo será cierta cuando la máquina de Turing determinista llegue a una configuración de aceptación o, dicho de otro modo, tenga alguna computación de aceptación. La implicación de

$$M_1 \text{ tiene computación de aceptación} \rightarrow \varphi_1 \text{ es satisfactible}$$

es clara. Veamos la contraria, es decir, veamos que si φ_1 es satisfactible, entonces M_1 tiene alguna computación de aceptación. Supongamos que

$$\varphi_1 \text{ es satisfactible} \not\rightarrow M_1 \text{ tiene computación de aceptación}$$

Si φ_1 es satisfactible, en particular la cláusula dentro de la subfórmula (2) $A(|u|^k, q_y)$ también es satisfactible. Supongamos que M_1 **no** tiene ninguna computación de aceptación. En tal caso, en ninguna de las computaciones llegaríamos a una configuración de aceptación; es decir, en ninguna de las computaciones alcanzaríamos una configuración cuyo estado sea q_y. En tal caso, la fórmula generada a partir de M_1 **no** sería satisfactible (debido a lo que se ha descrito más arriba). Hemos llegado una contradicción, por lo que queda demostrada la segunda implicación.

■

Vamos a ver cómo se construye la fórmula φ_1 a partir de M_1

1. **Configuración inicial:** $A(0, q_0) \wedge B(0, 1) \wedge C(0, 1, u) \wedge C(0, 2, B)$ (En este caso, la u depende del dato de entrada, ya hablaremos más adelante de esto).

2. **Configuración de aceptación:** $A(1, q_y) \wedge B(1, 1) \wedge C(1, 1, B) \wedge C(1, 2, B)$

3. M_1 **está en un solo estado:**

   $(A(0, q_0) \vee A(0, q_y) \vee A(0, q_n)) \wedge [(\neg A(0, q_0)) \vee (\neg A(0, q_y))] \wedge [(\neg A(0, q_0)) \vee (\neg A(0, q_n))] \wedge [(\neg A(0, q_y)) \vee (\neg A(0, q_n))] \wedge$
   $(A(1, q_0) \vee A(1, q_y) \vee A(1, q_n)) \wedge [(\neg A(1, q_0)) \vee (\neg A(1, q_y))] \wedge [(\neg A(1, q_0)) \vee (\neg A(1, q_n))] \wedge [(\neg A(1, q_y)) \vee (\neg A(1, q_n))]$

4. **Cada casilla tiene un solo símbolo:**

   $(C(0,1,0) \vee C(0,1,1) \vee C(0,1,B)) \wedge [(\neg C(0,1,0)) \vee (\neg C(0,1,1))] \wedge [(\neg C(0,1,0)) \vee (\neg C(0,1,B))] \wedge [(\neg C(0,1,1)) \vee (\neg C(0,1,B))] \wedge$
   $(C(1,1,0) \vee C(1,1,1) \vee C(1,1,B)) \wedge [(\neg C(1,1,0)) \vee (\neg C(1,1,1))] \wedge [(\neg C(1,1,0)) \vee (\neg C(1,1,B))] \wedge [(\neg C(1,1,1)) \vee (\neg C(1,1,B))] \wedge$
   $(C(0,2,0) \vee C(0,2,1) \vee C(0,2,B)) \wedge [(\neg C(0,2,0)) \vee (\neg C(0,2,1))] \wedge [(\neg C(0,2,0)) \vee (\neg C(0,2,B))] \wedge [(\neg C(0,2,1)) \vee (\neg C(0,2,B))] \wedge$
   $(C(1,2,0) \vee C(1,2,1) \vee C(1,2,B)) \wedge [(\neg C(1,2,0)) \vee (\neg C(1,2,1))] \wedge [(\neg C(1,2,0)) \vee (\neg C(1,2,B))] \wedge [(\neg C(1,2,1)) \vee (\neg C(1,2,B))]$

5. **El cabezal analiza una sola casilla:**

   $(B(0,1) \vee B(0,2)) \wedge [(\neg B(0,1)) \vee (\neg B(0,2))] \wedge$
   $(B(1,1) \vee B(1,2)) \wedge [(\neg B(1,1)) \vee (\neg B(1,2))]$

6. **La función de transición pasa de $\mathcal{C}_i$ a $\mathcal{C}_{i+1}$:** y

7. **Si una casilla no se analiza conserva su símbolo:** las construiremos juntas, debido a que vamos a construir, directamente, $\bigwedge_{0 \leq i < |u|^k, 1 \leq j \leq |u|^k} (6) \wedge$ (7) [1]

   $^{((1)_1)}[(A(0,q_0) \wedge B(0,1) \wedge C(0,1,0)) \rightarrow (A(1,q_n) \wedge B(1,1) \wedge C(1,1,0))] \wedge [(A(0,q_0) \wedge B(0,1) \wedge C(0,1,1)) \rightarrow (A(1,q_y) \wedge B(1,1) \wedge C(1,1,B))] \wedge [(A(0,q_0) \wedge B(0,1) \wedge C(0,1,B)) \rightarrow (A(1,q_0) \wedge B(1,1) \wedge C(1,1,B))] \wedge$
   $^{((2)_1)}[(C(0,1,0) \wedge \neg B(0,1) \wedge C(1,1,0)) \vee (C(0,1,1) \wedge \neg B(0,1) \wedge C(1,1,1)) \vee (C(0,1,B) \wedge \neg B(0,1) \wedge C(1,1,B))] \wedge$
   $^{((1)_2)}[(A(0,q_0) \wedge B(0,2) \wedge C(0,2,0)) \rightarrow (A(1,q_n) \wedge B(1,2) \wedge C(1,2,0))] \wedge [(A(0,q_0) \wedge B(0,2) \wedge C(0,2,1)) \rightarrow (A(1,q_y) \wedge B(1,2) \wedge C(1,2,B))] \wedge [(A(0,q_0) \wedge B(0,2) \wedge C(0,2,B)) \rightarrow (A(1,q_0) \wedge B(1,2) \wedge C(1,2,B))] \wedge$
   $^{((2)_2)}[(C(0,2,0) \wedge \neg B(0,2) \wedge C(1,2,0)) \vee (C(0,2,1) \wedge \neg B(0,2) \wedge C(1,2,1)) \vee (C(0,2,B) \wedge \neg B(0,2) \wedge C(1,2,B))]$

Vamos a ver que la fórmula φ_1 es satisfactible si y sólo si M_1 tiene una computación (en este caso, única) de aceptación. Para ello, nos tenemos que fijar en que la única subfórmula que cambia dependiendo de la entrada es la subfórmula (1). En particular, sólo el tercer literal $C(0,1,u)$ cambia dependiendo de la entrada u, que puede ser 0 o 1.

Veamos los dos casos, primero el caso $u = 0$ y el segundo $u = 1$. Cada configuración cumple una serie de restricciones, así que podemos pensar que, si cumple esas restricciones, los literales correspondientes tienen que ser **verdaderos**. Usamos $\varphi_{i,j}$ la fórmula generada a partir de la máquina de Turing no determinista M_i con dato de entrada $u = j$.

Para el caso negativo, tenemos la computación mostrada en la Figura 5. En dicho caso, cada configuración representa los literales que se hacen verdaderos.

[1] Hay que observar que en las diapositivas aparece como $1 \leq i \leq |u|^k$ pero por simplificar la creación de la fórmula, es suficiente con hacerlo hasta la penúltima configuración, dado que no hay una configuración siguiente a la última

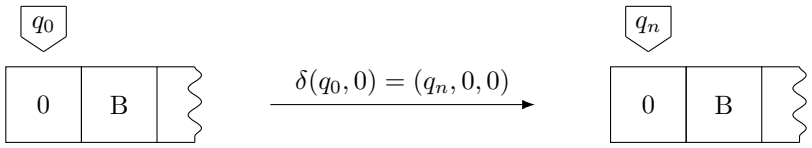

Figura 5: Computación de M_1 con dato de entrada $u = 0$

De hecho, dada la configuración $\mathcal{C}_t$ y la configuración $\mathcal{C}_{t+1}$, es <u>única</u> la transición que ha hecho posible el paso de $\mathcal{C}_t$ a $\mathcal{C}_{t+1}$. En particular, cada elemento hace verdadero los siguientes literales:

- **Configuración inicial**
  - *Estado*
    - $A(0, q_0), \neg A(0, q_y), \neg A(0, q_n)$
  - *Cabezal*
    - $B(0, 1), \neg B(0, 2)$
  - *Cinta*
    - $C(0, 1, 0), \neg C(0, 1, 1), \neg C(0, 1, B), C(0, 2, B), \neg C(0, 2, 0), \neg C(0, 2, 1)$
- **Configuración final**
  - *Estado*
    - $A(1, q_n), \neg A(1, q_0), \neg A(1, q_y)$
  - *Cabezal*
    - $B(1, 1), \neg B(1, 2)$
  - *Cinta*
    - $C(1, 1, 0), \neg C(1, 1, 1), \neg C(1, 1, B), C(1, 2, B), \neg C(1, 2, 0), \neg C(1, 2, 1)$

Para hacerlo fácil, podemos fijarnos en que, en la única computación de la máquina M_1 con dato de entrada $u = 0$, el estado q_y **nunca se alcanza**. En tal caso, el literal $A(1, q_y)$ asociado a la subfórmula (2) **nunca será satisfecho** (de hecho, como se ve en el *estado* de la **configuración final**, se hace verdadero su complementario, $\neg A(1, q_y)$). Por lo tanto, la fórmula $\varphi_{1,0}$ **no es satisfactible**.

Para el caso afirmativo, la máquina M_1 con dato de entrada $u = 1$ sigue una única computación representada en la Figura 6

Los literales que se hacen verdaderos por las configuraciones de la máquina son muy similares a la anterior, sólo cambiando algunos (marcados en negrita):

- **Configuración inicial**
  - *Estado*
    - $A(0, q_0), \neg A(0, q_y), \neg A(0, q_n)$

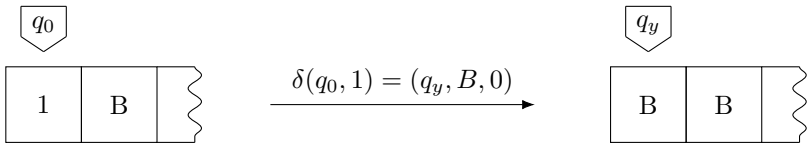

Figura 6: Computación de M_1 con dato de entrada $u = 1$

- *Cabezal*
  - $B(0,1), \neg B(0,2)$
- *Cinta*
  - $\mathbf{C(0,1,1)}, \neg\mathbf{C(0,1,0)}, \neg C(0,1,B), C(0,2,B), \neg C(0,2,0), \neg C(0,2,1)$

▪ **Configuración final**

- *Estado*
  - $\mathbf{A(1,q_y)}, \neg A(1,q_0), \neg\mathbf{A(1,q_n)}$
- *Cabezal*
  - $B(1,1), \neg B(1,2)$
- *Cinta*
  - $\mathbf{C(1,1,B)}, \neg\mathbf{C(1,1,0)}, \neg C(1,1,1), C(1,2,B), \neg C(1,2,0), \neg C(1,2,1)$

Vamos a ir fórmula a fórmula viendo cuáles se hacen verdaderas:

1. $A(0,q_0) \wedge B(0,1) \wedge C(0,1,1) \wedge C(0,1,B) \wedge C(0,2,B)$: todos los literales se hacen verdadero respecto a los literales indicados como **verdaderos** arriba (como era obvio).

2. $A(1,q_y) \wedge B(1,1) \wedge C(1,1,B) \wedge C(1,2,B)$: precisamente aquí podemos observar como la asignación **verdadera** que le da el estado de la **configuración final** al literal $A(1,q_y)$ es uno de los elementos (además del literal $C(1,1,B)$) que faltaba en el caso anterior para hacer verdadera esta subfórmula.

3. En este caso, vamos a marcar en azul los literales que son ciertos y en rojo los literales cuyos complementarios son verdaderos:

   $(\mathbf{A(0,q_0)} \vee \mathbf{A(0,q_y)} \vee \mathbf{A(0,q_n)}) \wedge [(\neg\mathbf{A(0,q_0)}) \vee (\neg\mathbf{A(0,q_y)})] \wedge [(\neg\mathbf{A(0,q_0)}) \vee (\neg\mathbf{A(0,q_n)})] \wedge [(\neg\mathbf{A(0,q_y)}) \vee (\neg\mathbf{A(0,q_n)})] \wedge$
   $(\mathbf{A(1,q_0)} \vee \mathbf{A(1,q_y)} \vee \mathbf{A(1,q_n)}) \wedge [(\neg\mathbf{A(1,q_0)}) \vee (\neg\mathbf{A(1,q_y)})] \wedge [(\neg\mathbf{A(1,q_0)}) \vee (\neg\mathbf{A(1,q_n)})] \wedge [(\neg\mathbf{A(1,q_y)}) \vee (\neg\mathbf{A(1,q_n)})]$

   Si se realizan los cálculos correspondientes, se puede comprobar que el valor de esta subfórmula es **verdadero**.

4. Al igual que en el caso anterior, vamos a marcar en azul los literales que se hacen verdaderos por alguna de las especificaciones de la computación, y en rojo los literales cuyos complementarios son marcados como verdaderos en la computación:

   $(\mathbf{C(0,1,0)} \vee \mathbf{C(0,1,1)} \vee \mathbf{C(0,1,B)}) \wedge [(\neg\mathbf{C(0,1,0)}) \vee (\neg\mathbf{C(0,1,1)})] \wedge$
   $[(\neg\mathbf{C(0,1,0)}) \vee (\neg\mathbf{C(0,1,B)})] \wedge [(\neg\mathbf{C(0,1,1)}) \vee (\neg\mathbf{C(0,1,B)})] \wedge$
   $(\mathbf{C(1,1,0)} \vee \mathbf{C(1,1,1)} \vee \mathbf{C(1,1,B)}) \wedge [(\neg\mathbf{C(1,1,0)}) \vee (\neg\mathbf{C(1,1,1)})] \wedge$
   $[(\neg\mathbf{C(1,1,0)}) \vee (\neg\mathbf{C(1,1,B)})] \wedge [(\neg\mathbf{C(1,1,1)}) \vee (\neg\mathbf{C(1,1,B)})] \wedge$
   $(\mathbf{C(0,2,0)} \vee \mathbf{C(0,2,1)} \vee \mathbf{C(0,2,B)}) \wedge [(\neg\mathbf{C(0,2,0)}) \vee (\neg\mathbf{C(0,2,1)})] \wedge$
   $[(\neg\mathbf{C(0,2,0)}) \vee (\neg\mathbf{C(0,2,B)})] \wedge [(\neg\mathbf{C(0,2,1)}) \vee (\neg\mathbf{C(0,2,B)})] \wedge$
   $(\mathbf{C(1,2,0)} \vee \mathbf{C(1,2,1)} \vee \mathbf{C(1,2,B)}) \wedge [(\neg\mathbf{C(1,2,0)}) \vee (\neg\mathbf{C(1,2,1)})] \wedge$
   $[(\neg\mathbf{C(1,2,0)}) \vee (\neg\mathbf{C(1,2,B)})] \wedge [(\neg\mathbf{C(1,2,1)}) \vee (\neg\mathbf{C(1,2,B)})]$

   Al igual que en el caso anterior, podemos comprobar que la subfórmula es **verdadera** con esta valoración de verdad, dada de forma implícita por la computación (en este caso, <u>única</u>) de la máquina M_1 con dato de entada $u = 1$

5. De nuevo, vemos con los colores anteriores la valoración de verdad que hace verdadera esta subfórmula:

   $(\mathbf{B(0,1)} \vee \mathbf{B(0,2)}) \wedge [(\neg\mathbf{B(0,1)}) \vee (\neg\mathbf{B(0,2)})] \wedge$
   $(\mathbf{B(1,1)} \vee \mathbf{B(1,2)}) \wedge [(\neg\mathbf{B(1,1)}) \vee (\neg\mathbf{B(1,2)})]$

   Volvemos a insistir que, teniendo presente que la información la estamos recibiendo directamente de la computación de la máquina de Turing no determinista, estamos obteniendo una valoración de verdad que hace verdadera la fórmula $\varphi_{1,1}$ (aunque para nosotros no sea relevante *cuál* es la valoración de verdad, sino su *existencia*, pero eso que nos llevamos). Aunque, en realidad, es precisamente la computación (gracias a la función de transición), la que va mostrándonos el «camino» hacia la valoración de verdad que hace verdadera la fórmula $\varphi_{1,1}$.

6. En este caso, como hicimos en la explicitación de la fórmula φ_1, vamos a realizar la asignación de $\bigwedge_{0\leq i<|u|^k, 1\leq j\leq|u|^k}$ $(6) \wedge (7)$:

   $^{((1)_1)}[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,1)} \wedge \mathbf{C(0,1,0)}) \rightarrow (\mathbf{A(1,q_n)} \wedge \mathbf{B(1,1)} \wedge \mathbf{C(1,1,0)})] \wedge$
   $[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,1)} \wedge \mathbf{C(0,1,1)}) \rightarrow (\mathbf{A(1,q_y)} \wedge \mathbf{B(1,1)} \wedge \mathbf{C(1,1,B)})] \wedge$
   $[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,1)} \wedge \mathbf{C(0,1,B)}) \rightarrow (\mathbf{A(1,q_0)} \wedge \mathbf{B(1,1)} \wedge \mathbf{C(1,1,B)})] \wedge$
   $^{((2)_1)}[(\mathbf{C(0,1,0)} \wedge \neg\mathbf{B(0,1)} \rightarrow \mathbf{C(1,1,0)}) \wedge (\mathbf{C(0,1,1)} \wedge \neg\mathbf{B(0,1)} \rightarrow \mathbf{C(1,1,1)}) \wedge$
   $(\mathbf{C(0,1,B)} \wedge \neg\mathbf{B(0,1)} \rightarrow \mathbf{C(1,1,B)})] \wedge$
   $^{((1)_2)}[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,2)} \wedge \mathbf{C(0,2,0)}) \rightarrow (\mathbf{A(1,q_n)} \wedge \mathbf{B(1,2)} \wedge \mathbf{C(1,2,0)})] \wedge$
   $[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,2)} \wedge \mathbf{C(0,2,1)}) \rightarrow (\mathbf{A(1,q_y)} \wedge \mathbf{B(1,2)} \wedge \mathbf{C(1,2,B)})] \wedge$
   $[(\mathbf{A(0,q_0)} \wedge \mathbf{B(0,2)} \wedge \mathbf{C(0,2,B)}) \rightarrow (\mathbf{A(1,q_0)} \wedge \mathbf{B(1,2)} \wedge \mathbf{C(1,2,B)})] \wedge$
   $^{((2)_2)}[(\mathbf{C(0,2,0)} \wedge \neg\mathbf{B(0,2)} \rightarrow \mathbf{C(1,2,0)}) \wedge (\mathbf{C(0,2,1)} \wedge \neg\mathbf{B(0,2)} \rightarrow \mathbf{C(1,2,1)}) \wedge$
   $(\mathbf{C(0,2,B)} \wedge \neg\mathbf{B(0,2)} \rightarrow \mathbf{C(1,2,B)})]$

   Es sencillo aplicar la valoración de verdad correspondiente para terminar obteniendo que esta subfórmula es **verdadera**. Téngase presente que una

implicación lógica es cierta si el *antecedente* es **falso** o el *consecuente* es verdadero

Todas las subfórmulas de $\varphi_{1,1}$ han sido satisfechas por la valoración de verdad que hemos extraído directamente de la única computación de M_1 con dato de entrada $u = 1$. Eso vendría a decir que todas las cláusulas en la forma normal conjuntiva serían verdaderas y, por tanto, la fórmula $\varphi_{1,1}$ tendría al menos una valoración de verdad que la hace verdadera; es decir, $\varphi_{1,1}$ es **satisfactible** (obvio, dado que M_1 con dato de entrada $u = 1$ tiene una computación de aceptación y $\varphi_{1,1}$ es la fórmula obtenida por la función $F : \Sigma_X^* \to E_{\texttt{SAT}}$).