Teoría de la Computación puesta en Práctica

Marcelo Arenas

Problema a resolver

WiMAX (Worldwide Interoperability for Microwave Access): estándar para comunicación inalámbrica

- Diseñado para funcionar en distancias más larga que Wi-Fi

Problema a resolver: Dar acceso a Internet a través de WiMAX a los habitantes de Juan Fernández.

Problema a resolver

Debemos instalar el menor número posible de antenas dando a cada casa un acceso a Internet con una velocidad de transmisión de datos mayor o igual a un cierto valor.

Es necesario considerar que:

- La comunicación se establece entre una antena y una casa, los datos trasmitidos a una casa no vienen de varias antenas
- La velocidad de transmisión de datos entre una antena y una casa depende de la distancia entre ellos
- Podemos suponer que una antena puede establecer comunicación con un número ilimitado de casas

Problema a resolver: formalización

Parámetros del problema:

- Casas a las cuales hay que dar conexión a Internet
- Conjunto de lugares posibles para las antenas
- Velocidad mínima permitida para la transmisión de datos entre las antenas y las casas

Problema a resolver: formalización

Entrada del problema: (P, C, t, v)

Problema a resolver: formalización

Entrada del problema: (P, C, t, v)

- P: conjunto de identificadores de posibles lugares para las antenas

Problema a resolver: formalización

Entrada del problema: (P, C, t, v)

- P: conjunto de identificadores de posibles lugares para las antenas
- C: conjunto de identificadores para las casas ($C \cap P = \emptyset$)

Problema a resolver: formalización

Entrada del problema: (P, C, t, v)

- P: conjunto de identificadores de posibles lugares para las antenas
- C: conjunto de identificadores para las casas ($C \cap P = \emptyset$)
- t : función que indica la velocidad de transmisión desde cada antena a cada casa

$$t \quad : \quad P \times C \rightarrow (\mathbb{R}^+ \cup \{0\})$$

Problema a resolver: formalización

Entrada del problema: (P, C, t, v)

- P: conjunto de identificadores de posibles lugares para las antenas
- C: conjunto de identificadores para las casas ($C \cap P = \emptyset$)
- t : función que indica la velocidad de transmisión desde cada antena a cada casa

$$t \quad : \quad P \times C \rightarrow (\mathbb{R}^+ \cup \{0\})$$

- v : velocidad mínima permitida para la transmisión de datos entre las antenas y las casas

Problema a resolver: formalización

Distribución factible para la entrada (P, C, t, v): $P' \subseteq P$

para todo $c \in C$, existe $p \in P'$: $t(p, c) \geq v$

Problema a resolver: formalización

Distribución factible para la entrada (P, C, t, v): $P' \subseteq P$

para todo $c \in C$, existe $p \in P'$: $t(p, c) \geq v$

Solución óptima para la entrada (P, C, t, v): $k \leq |P|$

existe distribución factible P' para (P, C, t, v) tal que $k = |P'|$, y para toda distribución factible P'' para (P, C, t, v) se tiene que $k \leq |P''|$

Problema a resolver: versión de decisión

Problema de decisión:

Entrada	:	(P, C, t, v, k)
Pregunta	:	¿existe una distribución factible P' para (P, C, t, v) tal que $\|P'\| = k$?

Problema a resolver: versión de decisión

Problema de decisión:

Entrada	:	(P, C, t, v, k)
Pregunta	:	¿existe una distribución factible P' para (P, C, t, v) tal que $\|P'\| = k$?

¿Es el problema de optimización más difícil que el problema de decisión?

Problema a resolver: versión de decisión

Problema de decisión:

Entrada	:	(P, C, t, v, k)
Pregunta	:	¿existe una distribución factible P' para (P, C, t, v) tal que $\|P'\| = k$?

¿Es el problema de optimización más difícil que el problema de decisión?

Ejercicio

Demuestre que si puede resolver uno de los problemas en tiempo polinomial, entonces puede resolver el otro en tiempo polinomial

Problema a resolver: versión de decisión

Problema de decisión:

Entrada	:	(P, C, t, v, k)
Pregunta	:	¿existe una distribución factible P' para (P, C, t, v) tal que $\|P'\| = k$?

¿Es el problema de optimización más difícil que el problema de decisión?

Ejercicio

Demuestre que si puede resolver uno de los problemas en tiempo polinomial, entonces puede resolver el otro en tiempo polinomial

Nos vamos a concentrar entonces en el problema de decisión, al que llamaremos WM.

El problema WM

¿Puede dar un algoritmo eficiente (de tiempo polinomial) para resolver WM?

- El representar WM como un problema sobre grafos podría ayudar . . .

El problema WM

¿Puede dar un algoritmo eficiente (de tiempo polinomial) para resolver WM?

- El representar WM como un problema sobre grafos podría ayudar . . .

No es obvio que exista tal algoritmo.

El problema WM

¿Puede dar un algoritmo eficiente (de tiempo polinomial) para resolver WM?

- El representar WM como un problema sobre grafos podría ayudar . . .

No es obvio que exista tal algoritmo.

- ¿Podemos demostrar que WM $\notin$ PTIME?

El problema WM

¿Puede dar un algoritmo eficiente (de tiempo polinomial) para resolver WM?

- El representar WM como un problema sobre grafos podría ayudar . . .

No es obvio que exista tal algoritmo.

- ¿Podemos demostrar que WM $\notin$ PTIME?
- ¿Podemos al menos dar evidencia sobre esto?

El problema WM

¿Puede dar un algoritmo eficiente (de tiempo polinomial) para resolver WM?

- El representar WM como un problema sobre grafos podría ayudar . . .

No es obvio que exista tal algoritmo.

- ¿Podemos demostrar que WM $\notin$ PTIME?
- ¿Podemos al menos dar evidencia sobre esto?
  - Para esto vamos a introducir la noción de reducción

La noción de reducción

Dado: Lenguajes L_1 y L_2 con alfabeto A.

Intuición: Decimos que un lenguaje L_1 es reducible a un lenguaje L_2 si existe un algoritmo tal que dado $w_1 \in A^*$, calcula $w_2 \in A^*$ tal que $w_1 \in L_1$ si y sólo si $w_2 \in L_2$.

Para formalizar esta idea necesitamos introducir la noción de algoritmo que calcula.

- Tenemos que introducir las máquinas de Turing que calculan.

Máquinas de Turing que calculan

Definición

Una MT M calcula una función $f : A^* \to A^*$ si:

> *con entrada $w \in A^*$, la máquina se detiene en un estado final teniendo en la memoria una cadena infinita de símbolos* B *seguida de $f(w)$ seguida de otra cadena infinita de símbolos* B*.*

Máquinas de Turing que calculan

Definición

Una MT M calcula una función $f : A^* \to A^*$ si:

> *con entrada $w \in A^*$, la máquina se detiene en un estado final teniendo en la memoria una cadena infinita de símbolos* B *seguida de $f(w)$ seguida de otra cadena infinita de símbolos* B*.*

Definición

Una función $f : A^* \to A^*$ es computable si es que existe una máquina de Turing que calcula f.

Máquinas de Turing que calculan

Ejemplo

Sea $g : \{0,1\}^* \rightarrow \{0,1\}^*$ una función definida como $g(w) = w0000$.
Queremos demostrar que g es computable.

Máquinas de Turing que calculan

Ejemplo

Sea $g : \{0,1\}^* \to \{0,1\}^*$ una función definida como $g(w) = w0000$.
Queremos demostrar que g es computable.

Sea $M = (Q, A, q_1, \delta, F)$, donde

- $Q = \{q_1, q_2, q_3, q_4, q_f\}$
- $A = \{0,1\}$
- $F = \{q_f\}$
- δ es definida como:

$$\begin{array}{lcl} \delta(q_1,0) & = & (q_1,0,\rightarrow) \\ \delta(q_1,1) & = & (q_1,1,\rightarrow) \\ \delta(q_1,\mathrm{B}) & = & (q_2,0,\rightarrow) \end{array} \qquad \begin{array}{l} \delta(q_2,\mathrm{B}) = (q_3,0,\rightarrow) \\ \delta(q_3,\mathrm{B}) = (q_4,0,\rightarrow) \\ \delta(q_4,\mathrm{B}) = (q_f,0,\rightarrow) \end{array}$$

La noción de reducción: formalización

Definición

Dados lenguajes L_1 y L_2 con alfabeto A, L_1 es reducible a L_2 si existe una función computable f tal que para todo $w \in A^*$: $w \in L_1$ si y sólo si $f(w) \in L_2$.

La noción de reducción: formalización

Definición

Dados lenguajes L_1 y L_2 con alfabeto A, L_1 es reducible a L_2 si existe una función computable f tal que para todo $w \in A^*$: $w \in L_1$ si y sólo si $f(w) \in L_2$.

Proposición

Suponga que L_1 es reducible a L_2.

- *Si L_2 es decidible entonces L_1 es decidible*
- *Si L_1 es indecidible entonces L_2 es indecidible*

La noción de reducción: formalización

Definición

Dados lenguajes L_1 y L_2 con alfabeto A, L_1 es reducible a L_2 si existe una función computable f tal que para todo $w \in A^*$: $w \in L_1$ si y sólo si $f(w) \in L_2$.

Proposición

Suponga que L_1 es reducible a L_2.

- *Si L_2 es decidible entonces L_1 es decidible*
- *Si L_1 es indecidible entonces L_2 es indecidible*

Pero nosotros necesitamos una noción de reducción más fuerte ...

La noción de reducción polinomial

Decimos que una función f es calculable en tiempo t si existe una MT M que calcula f y tal que t_M es $O(t)$.

La noción de reducción polinomial

Decimos que una función f es calculable en tiempo t si existe una MT M que calcula f y tal que t_M es $O(t)$.

Definición

Dados lenguajes L_1 y L_2 con alfabeto A, decimos que L_1 es reducible en tiempo polinomial a L_2 si existe una función f computable en tiempo polinomial tal que para todo $w \in A^*$:

$$w \in L_1 \text{ si y sólo si } f(w) \in L_2.$$

La noción de reducción polinomial

Proposición

Suponga que L_1 es reducible en tiempo polinomial a L_2.

- *Si $L_2 \in PTIME$ entonces $L_1 \in PTIME$*
- *Si $L_1 \notin PTIME$ entonces $L_2 \notin PTIME$*

Ejercicio

Demuestre la proposición.

Mostrando que un problema es difícil . . .

Para demostrar que WM $\notin$ PTIME, basta entonces encontrar un lenguaje $L \notin$ PTIME y dar una reducción polinomial de L a WM

Mostrando que un problema es difícil . . .

Para demostrar que WM $\notin$ PTIME, basta entonces encontrar un lenguaje $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje? ¿Podría ser H10?

Mostrando que un problema es difícil . . .

Para demostrar que WM $\notin$ PTIME, basta entonces encontrar un lenguaje $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje? ¿Podría ser H10?

Para dar evidencia que WM $\notin$ PTIME, basta encontrar un lenguaje L para el cual se cree que $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje?

Mostrando que un problema es difícil . . .

Para demostrar que WM $\notin$ PTIME, basta entonces encontrar un lenguaje $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje? ¿Podría ser H10?

Para dar evidencia que WM $\notin$ PTIME, basta encontrar un lenguaje L para el cual se cree que $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje?

Tenemos muchas alternativas para la última pregunta: 3-COL, SAT, programación lineal entera, el problema del agente viajero, . . .

Mostrando que un problema es difícil . . .

Para demostrar que WM $\notin$ PTIME, basta entonces encontrar un lenguaje $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje? ¿Podría ser H10?

Para dar evidencia que WM $\notin$ PTIME, basta encontrar un lenguaje L para el cual se cree que $L \notin$ PTIME y dar una reducción polinomial de L a WM

- ¿Cuál podría ser este lenguaje?

Tenemos muchas alternativas para la última pregunta: 3-COL, SAT, programación lineal entera, el problema del agente viajero, . . .

- ¿Por qué se cree que estos problemas no están en PTIME? ¿Cómo se comparan estos problemas en base a la noción de reducción polinomial?

WM es problema un problema difícil

Vamos a demostrar que SAT se puede reducir en tiempo polinomial a WM.

- Pero antes vamos a dar una (muy) breve introducción a SAT

WM es problema un problema difícil

Vamos a demostrar que SAT se puede reducir en tiempo polinomial a WM.

- Pero antes vamos a dar una (muy) breve introducción a SAT

Nociones importantes:

- Letra proposicional: p, q, r
- Conectivos proposicionales $\neg$, $\vee$, $\wedge$

Una (muy) breve introducción a SAT

- La semántica de los conectivos proposicionales es dada con una tabla de verdad:

p	q	$\neg p$	$p \vee q$	$p \wedge q$
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

- Literal: letra proposicional (p) o la negación de una letra proposicional ($\neg q$)
- Cláusula: disyunción de literales, por ejemplo $(p \vee \neg q \vee r)$

Una (muy) breve introducción a SAT

- Fórmula proposicional en CNF: conjunción de cláusulas, por ejemplo $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$
- Para cada fórmula proposicional en CNF asociamos una tabla de verdad:

p	q	r	$(p \vee \neg q \vee r)$	$(\neg p \vee q)$	$(p \vee \neg q \vee r) \wedge (\neg p \vee q)$
0	0	0	1	1	1
0	0	1	1	1	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	1

Una (muy) breve introducción a SAT

Finalmente podemos definir SAT:

$$\text{SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional en CNF tal que la tabla de verdad de } \varphi \text{ tiene una fila con un 1 asignado a } \varphi\}$$

Una (muy) breve introducción a SAT

Finalmente podemos definir SAT:

$$\text{SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional en CNF tal que la tabla de verdad de } \varphi \text{ tiene una fila con un 1 asignado a } \varphi\}$$

Ejercicio

De fórmulas proposicionales α, β en CNF tales que $\alpha \in \text{SAT}$ y $\beta \notin \text{SAT}$.

Una reducción polinomial de SAT a WM

Queremos dar un algoritmo polinomial que calcula lo siguiente:

Entrada : fórmula proposicional φ en CNF
Salida : (P, C, t, v, k) tal que $\varphi \in$ SAT si y sólo si $(P, C, t, v, k) \in$ WM

Una reducción polinomial de SAT a WM

Queremos dar un algoritmo polinomial que calcula lo siguiente:

Entrada : fórmula proposicional φ en CNF
Salida : (P, C, t, v, k) tal que $\varphi \in$ SAT si y sólo si $(P, C, t, v, k) \in$ WM

Vamos a ver cómo funciona este algoritmo con la fórmula proposicional $\alpha_1 \wedge \alpha_2$, donde

$$\alpha_1 = (p \vee \neg q \vee r)$$
$$\alpha_2 = (\neg p \vee q)$$

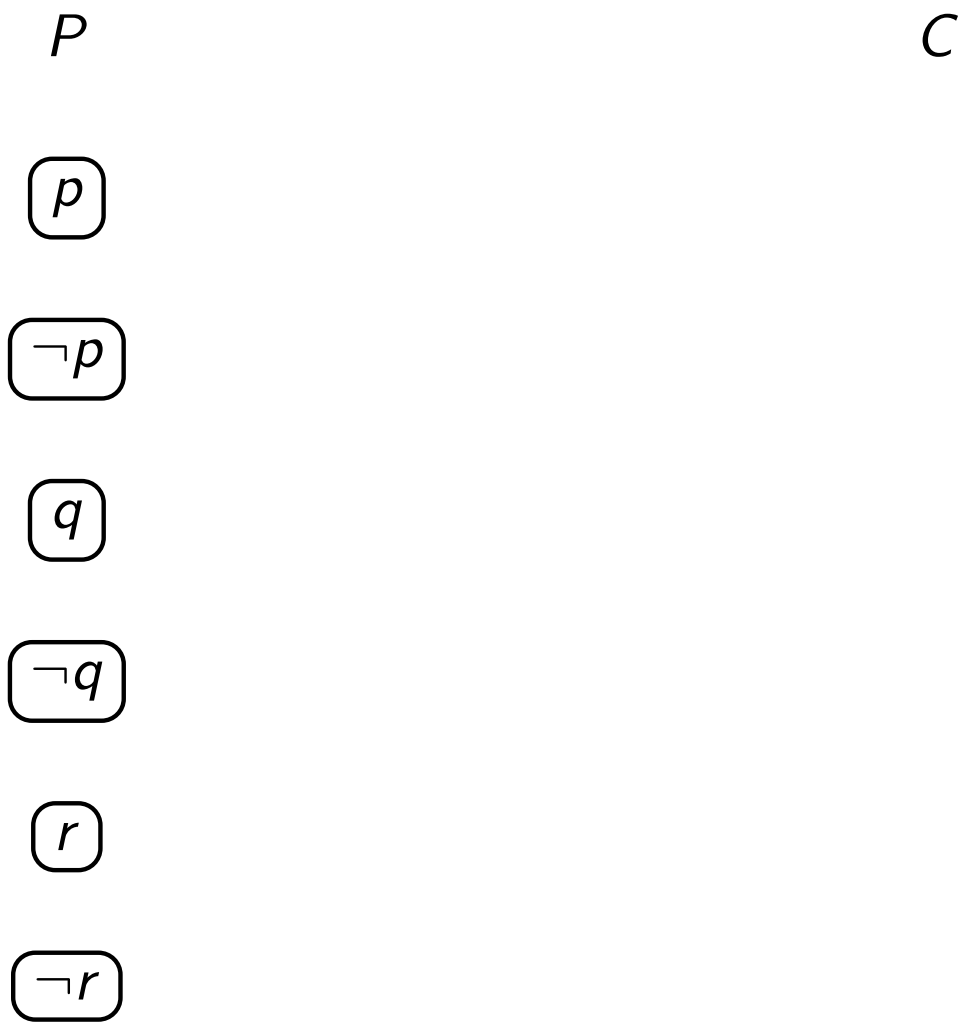
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

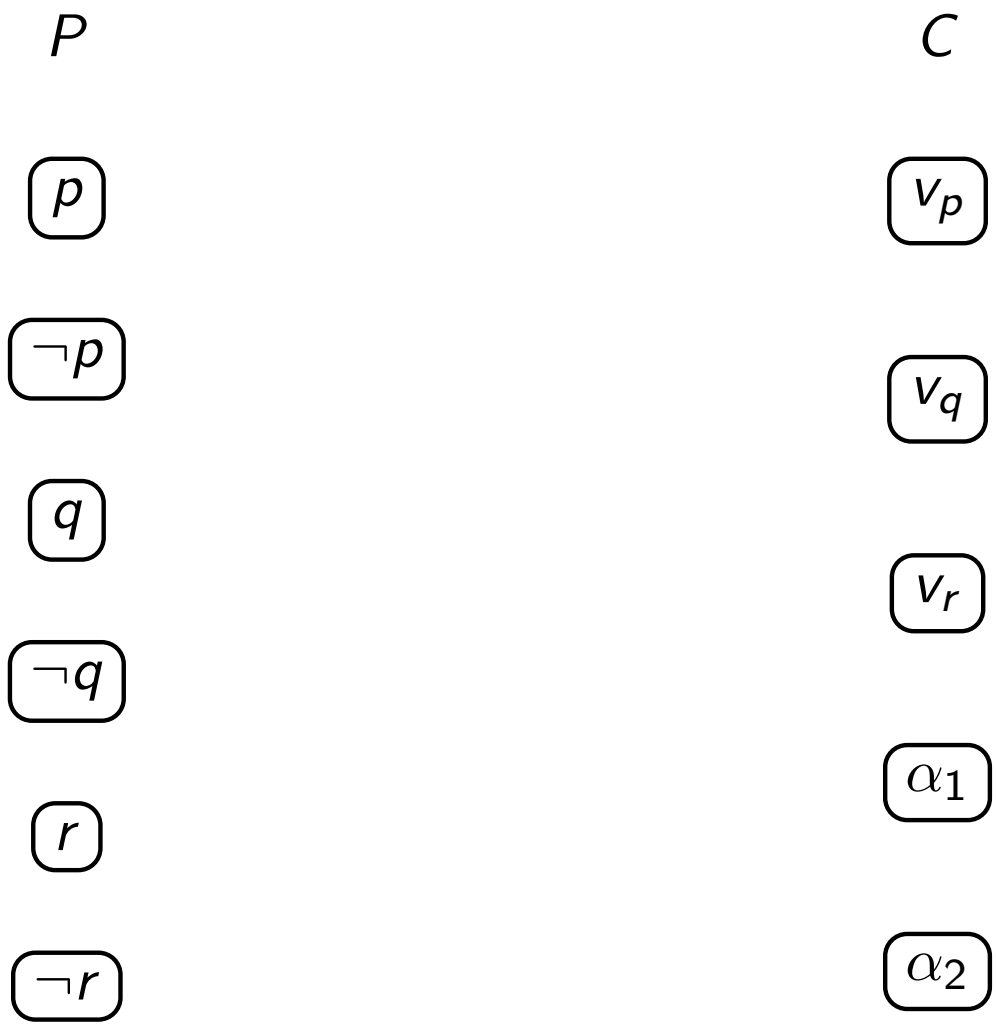
P

Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

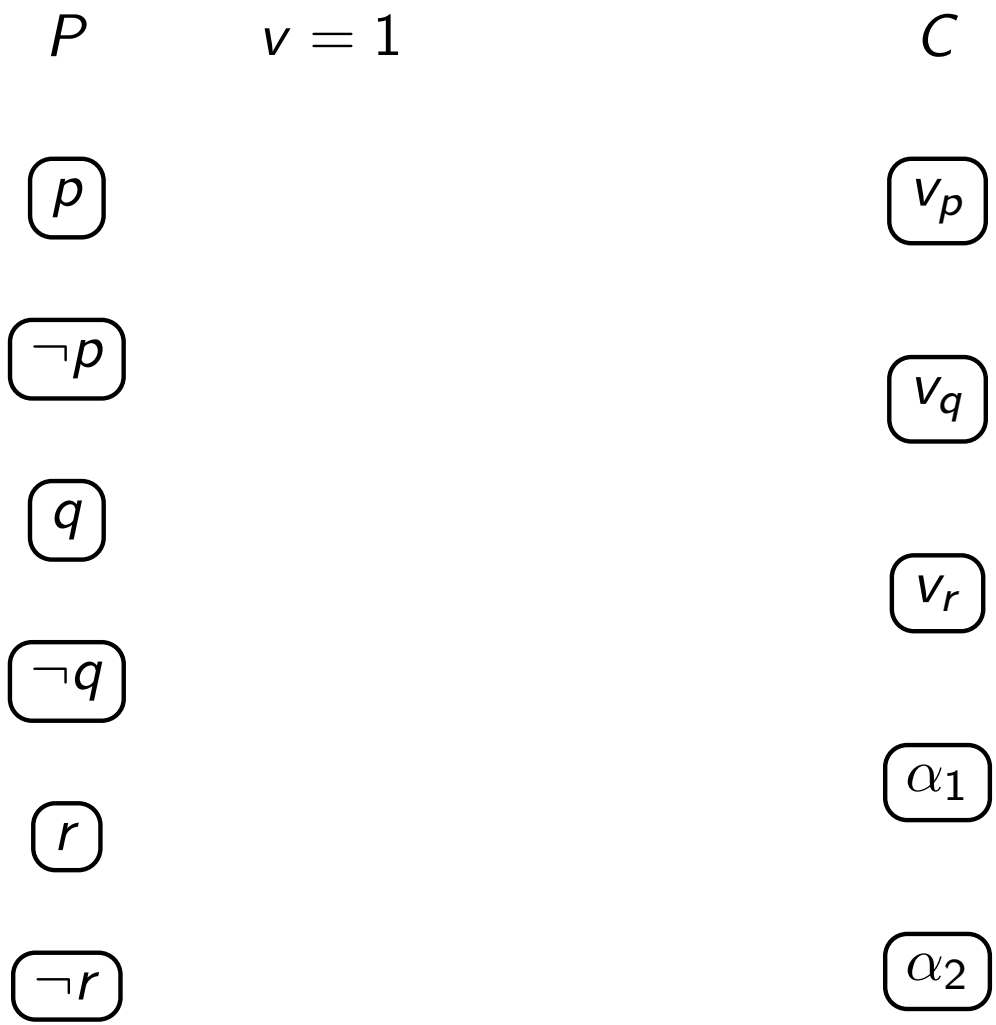
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



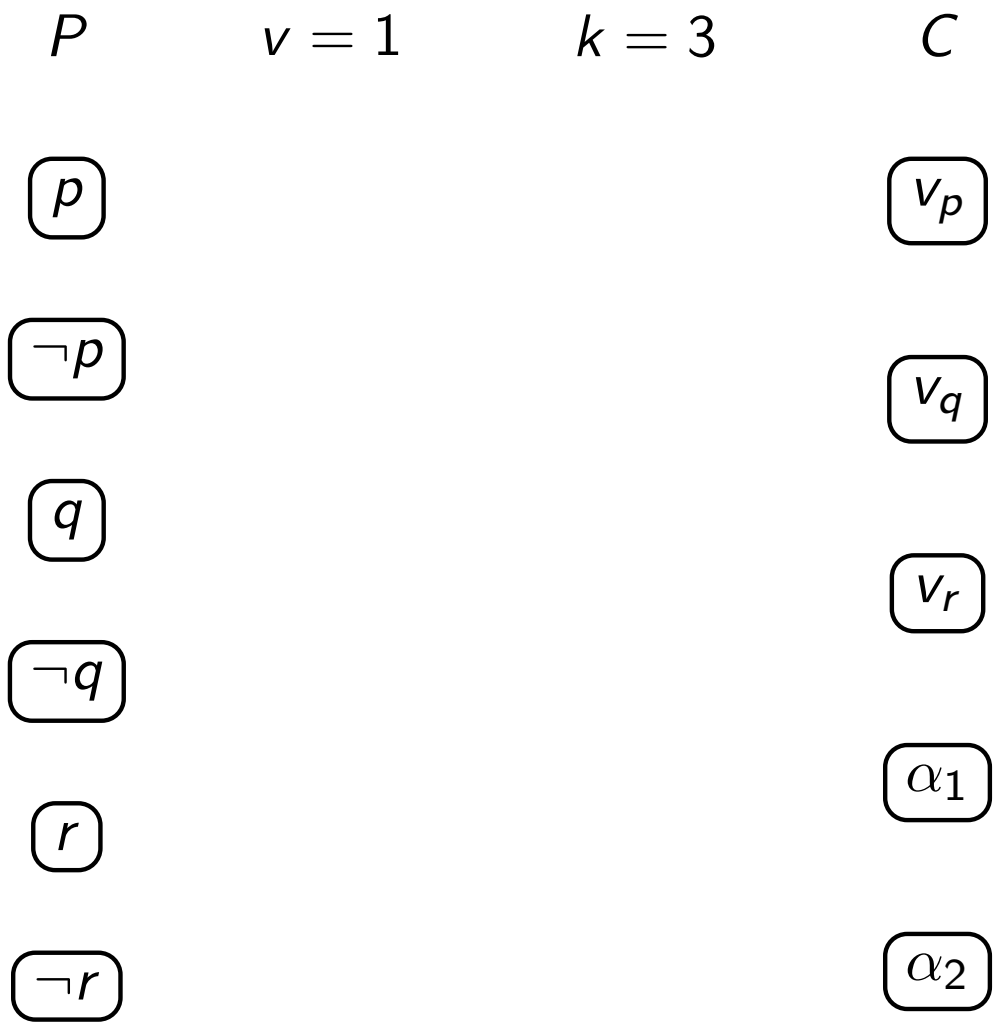
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



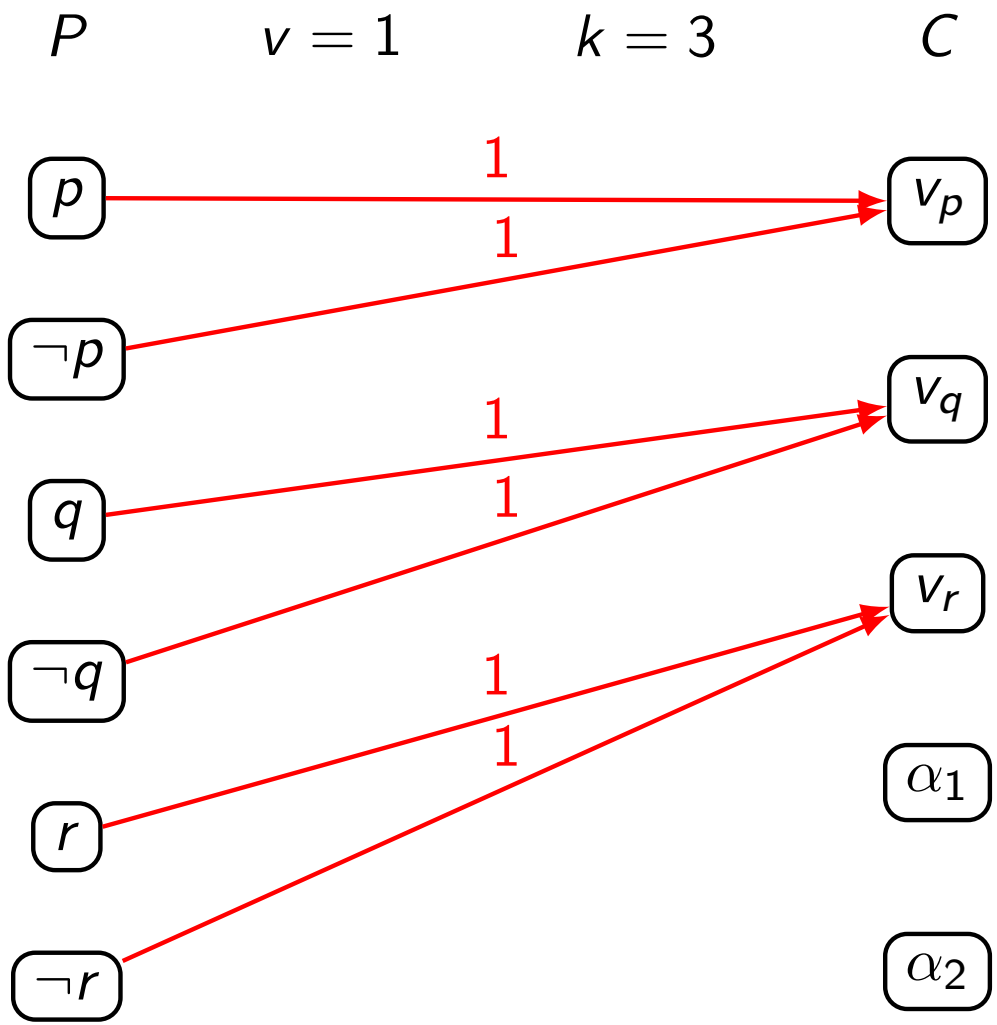
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



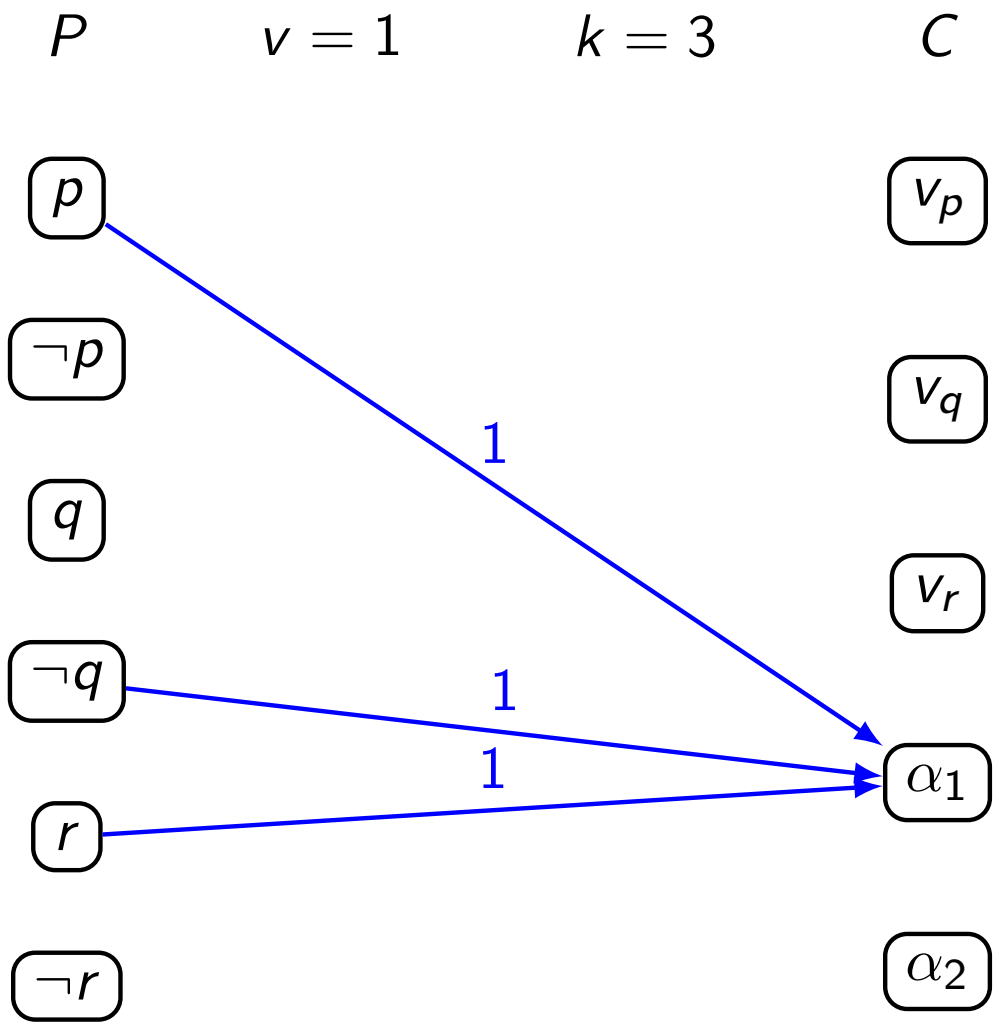
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



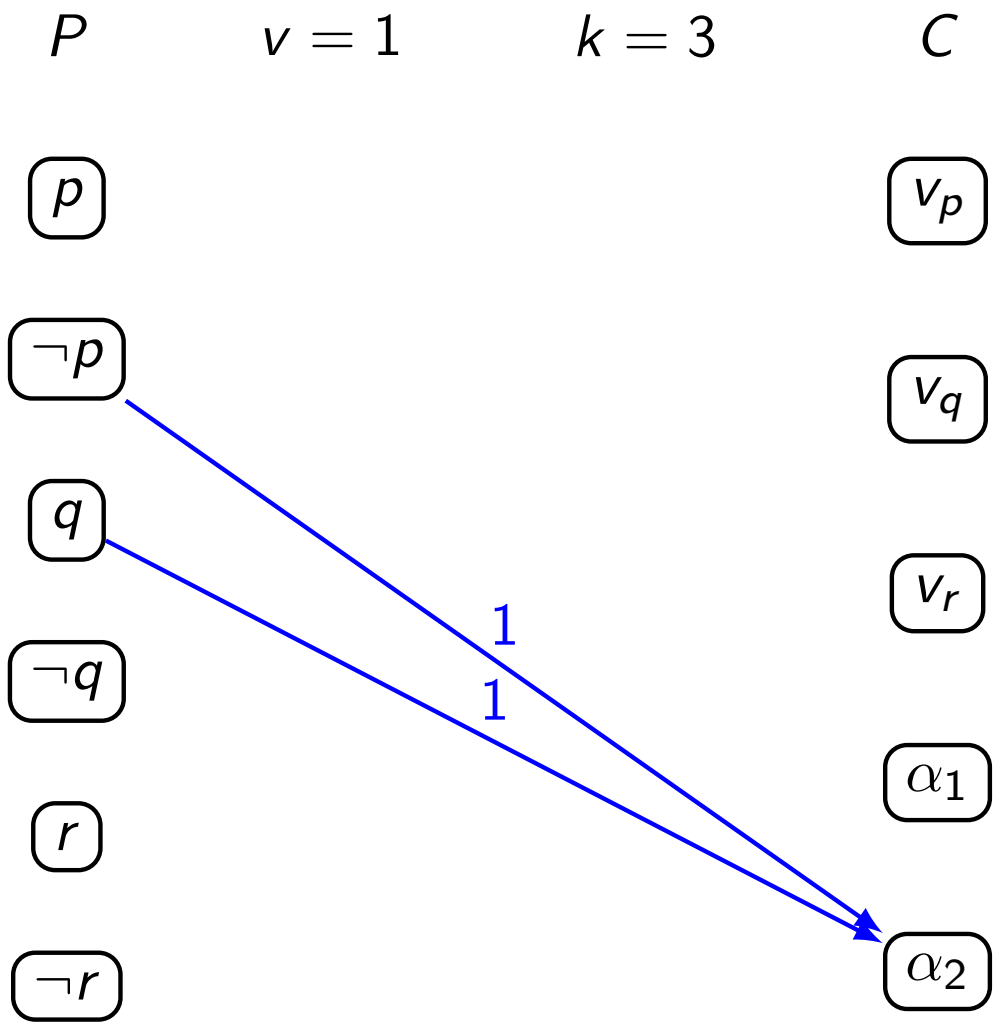
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



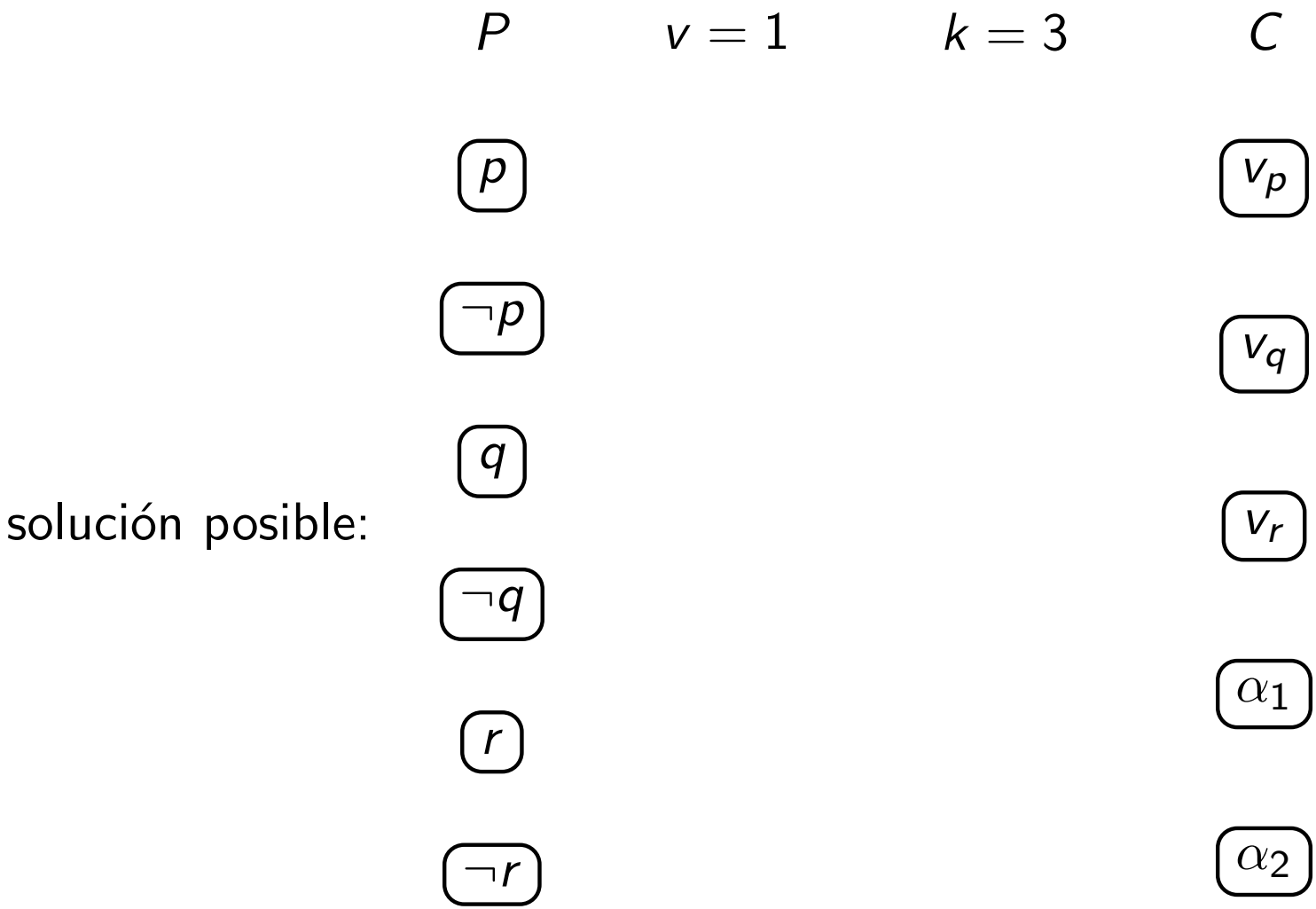
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



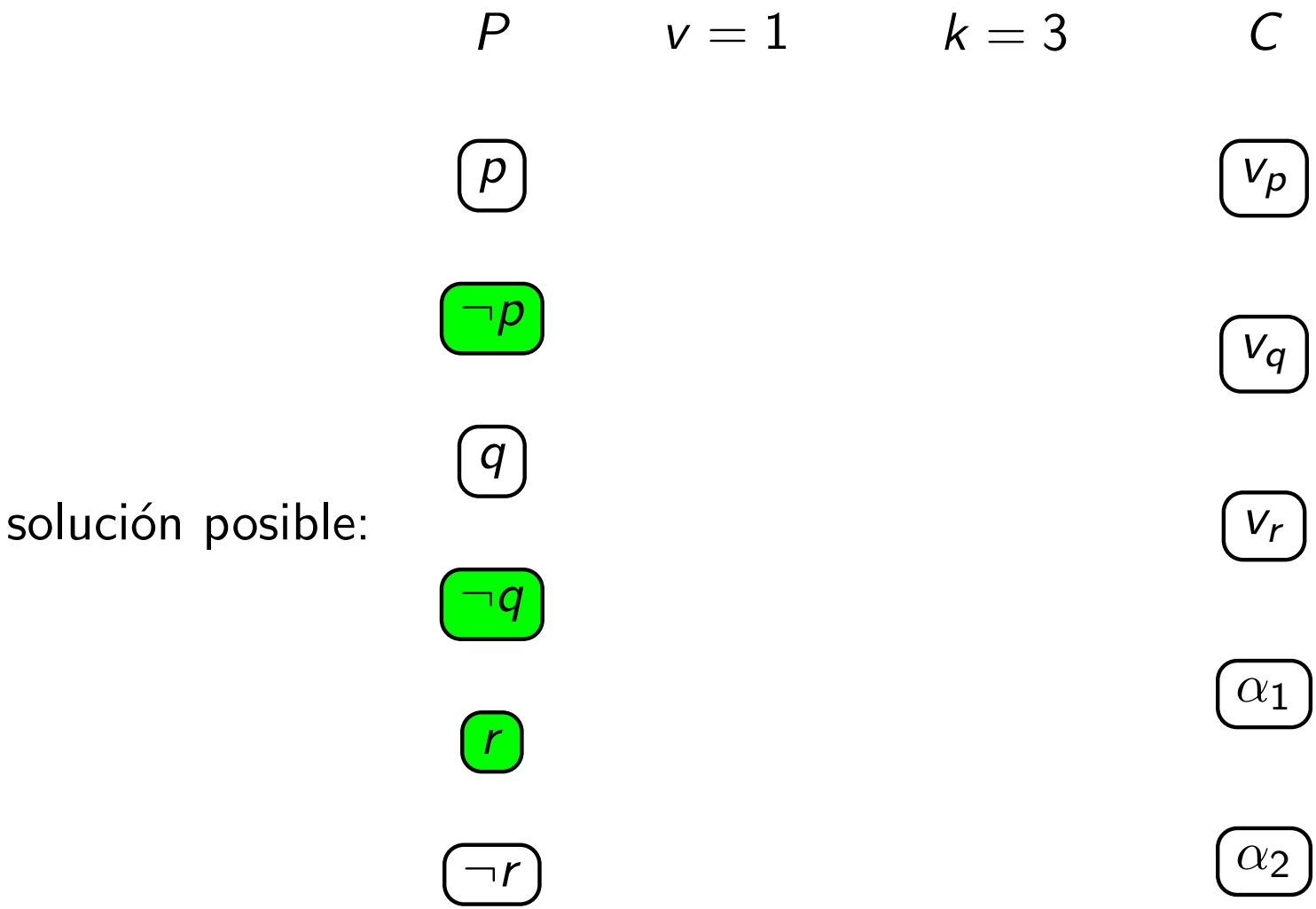
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

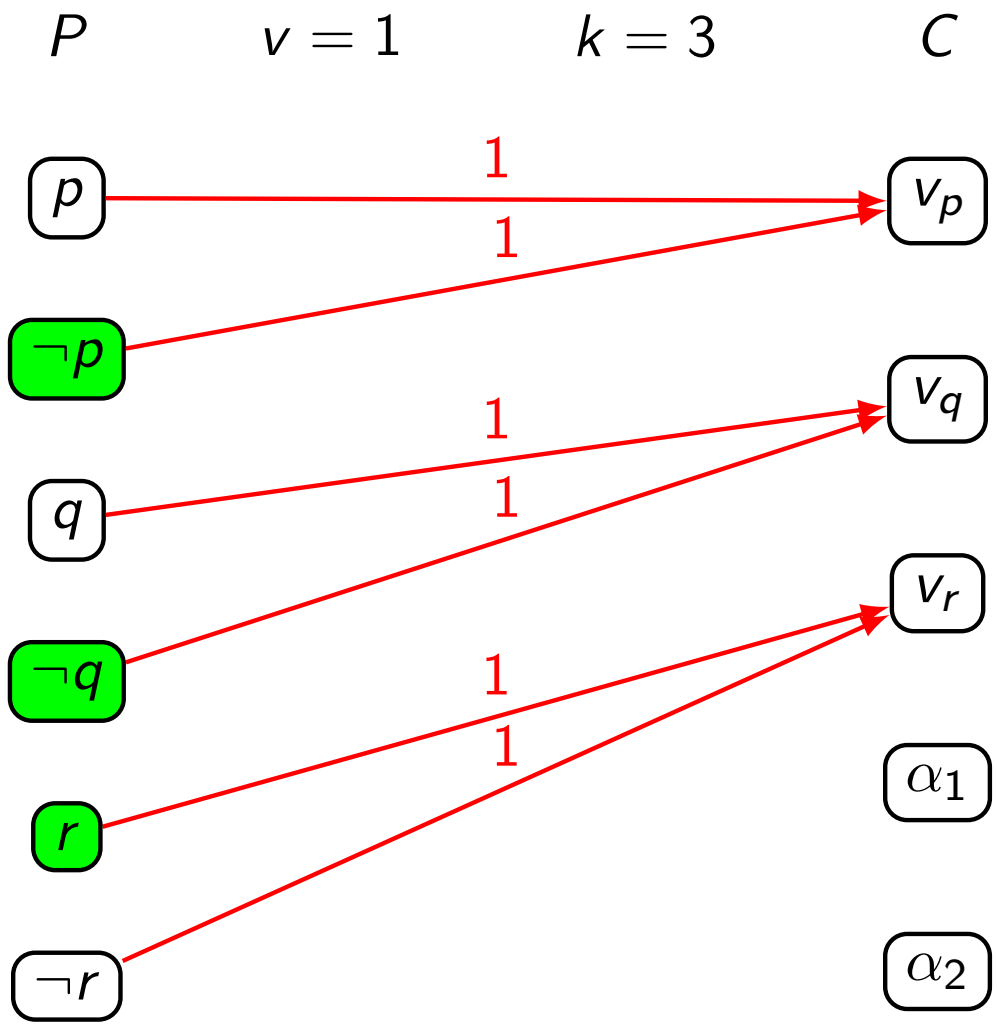


Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$



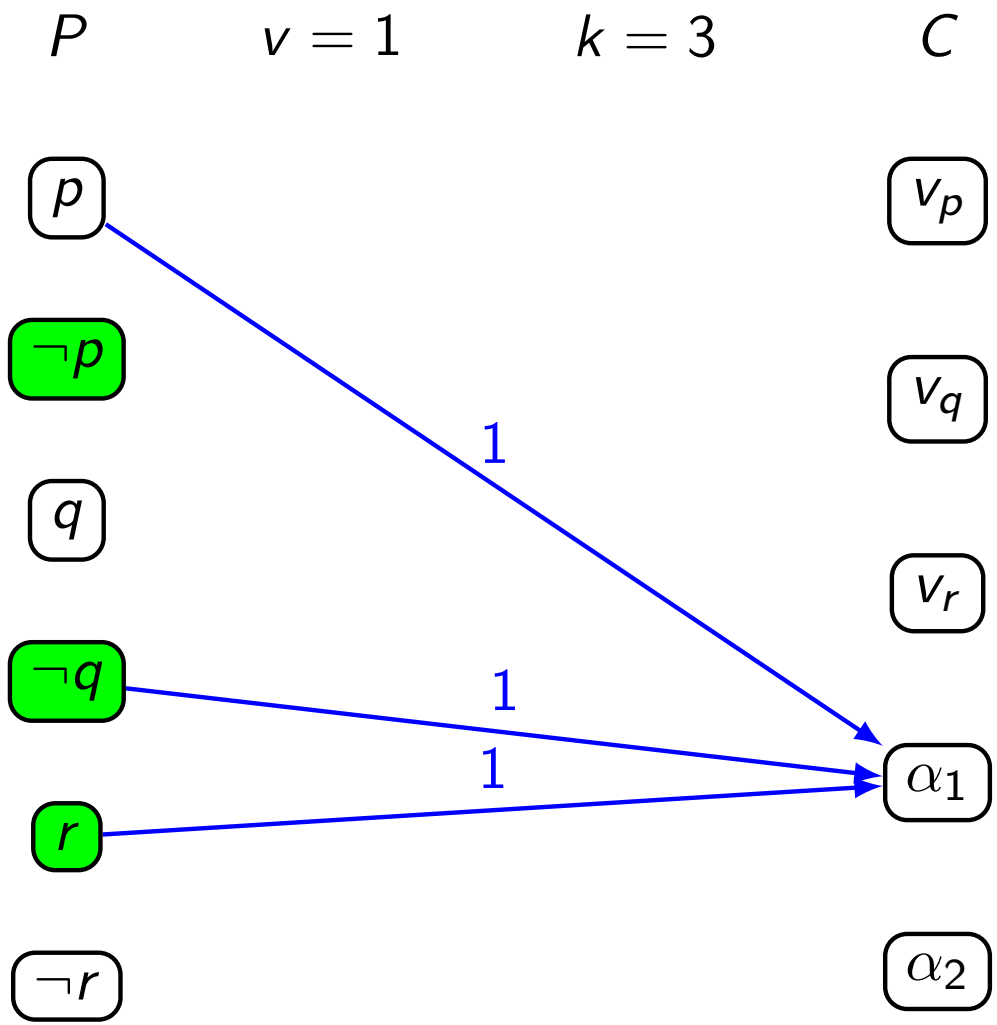
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

solución posible:



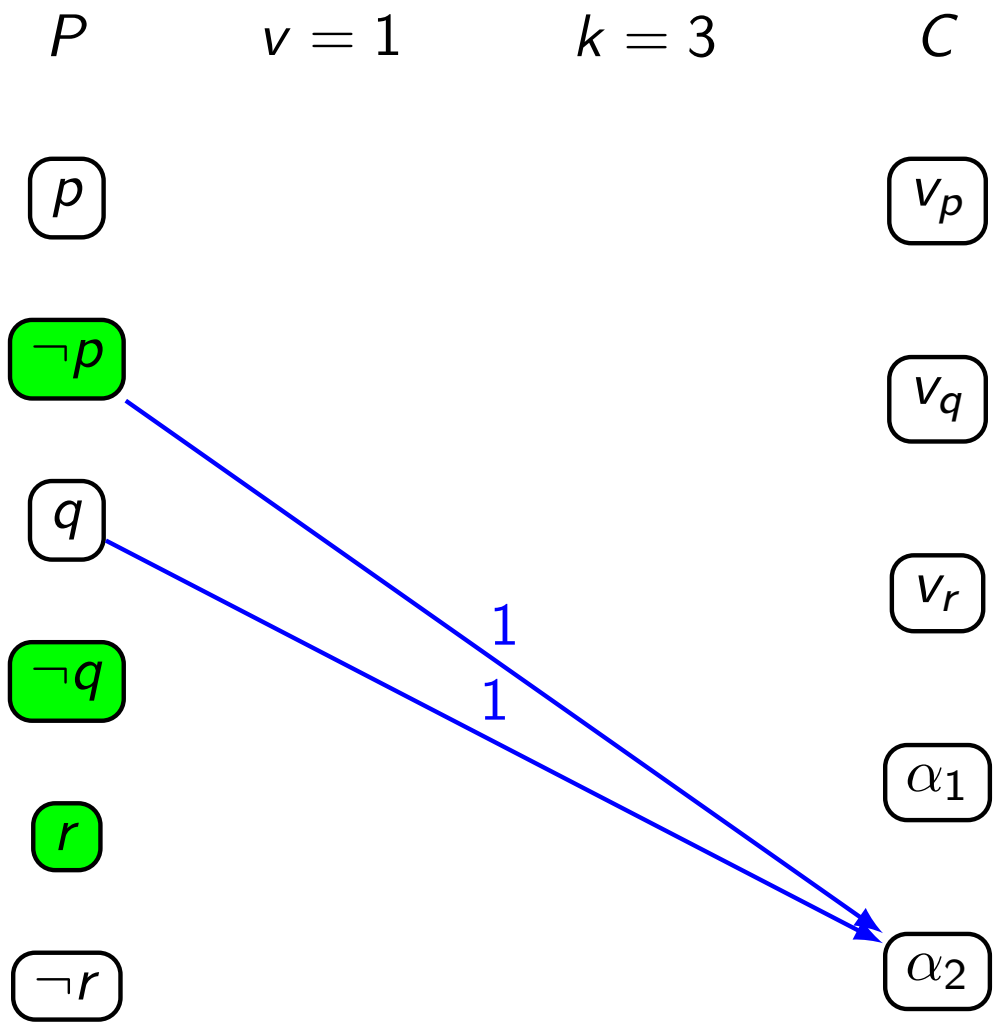
Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

solución posible:



Representando $(p \vee \neg q \vee r) \wedge (\neg p \vee q)$

solución posible:



Una reducción polinomial de SAT a WM: Ejercicios

1. Generalice la construcción anterior para cualquier fórmula proposicional φ en CNF

2. Demuestre que la reducción es correcta. Vale decir, suponiendo que φ es una fórmula proposicional en CNF y que la salida del algoritmo para φ es (P, C, t, v, k), demuestre que $\varphi \in$ SAT si y sólo si $(P, C, t, v, k) \in$ WM

3. Demuestre que la reducción puede ser calculada en tiempo polinomial

¿Cómo podemos solucionar WM?

Creemos que WM es un problema difícil, ¿nos damos por vencido entonces?

¿Cómo podemos solucionar WM?

Creemos que WM es un problema difícil, ¿nos damos por vencido entonces?

- No, usamos la noción de reducción nuevamente

¿Cómo podemos solucionar WM?

Creemos que WM es un problema difícil, ¿nos damos por vencido entonces?

- No, usamos la noción de reducción nuevamente

Existen buenos algoritmos para resolver SAT, programación lineal entera y el problema del agente viajero en la práctica.

- Funcionan en tiempo exponencial en el peor caso

¿Cómo podemos solucionar WM?

Podemos entonces reducir WM a uno de estos problemas y usar alguno de estos algoritmos.

¿Cómo podemos solucionar WM?

Podemos entonces reducir WM a uno de estos problemas y usar alguno de estos algoritmos.

Ejercicio

Muestre cómo reducir en tiempo polinomial WM a programación lineal entera.