eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Lenguajes, Computación y Sistemas Inteligentes

Grado en Ingeniería Informática de Gestión y Sistemas de Información

Escuela de Ingeniería de Bilbao (UPV/EHU)

2^o curso

Curso académico 2023-2024

Tema 2: Sistemas Inteligentes

JOSÉ GAINTZARAIN IBARMIA

Departamento de Lenguajes y Sistemas Informáticos

Última actualización: 04 - 09 - 2023

Índice general

Índice de figuras

Índice de tablas

Tema 2

Sistemas Inteligentes

2.1.

Introducción

Tal como se ha indicado en el Tema 1, llamamos Sistemas Inteligentes [1] a los sistemas que llevan a cabo tareas o cálculos que no admiten una especificación funcional. Por tanto, la descripción del resultado no se puede expresar utilizando solo los datos de entrada que se tienen al principio. En estos sistemas que no admiten especificación funcional, hay información adicional que llega más tarde, durante el proceso de cálculo. Esa información adicional no está disponible al principio del programa pero es imprescindible para describir el resultado que se obtendrá.

En la figura 2.1.1 de la página 13, se muestra el esquema general de un problema que no admite especificación funcional. Ahí, los datos de entrada son $x_1, x_2, \ldots, x_n$ pero el resultado $f(\overline{x}, \overline{y}, \overline{z})$ —donde $\overline{x} \equiv x_1, x_2, \ldots, x_n$, $\overline{y} \equiv y_1, y_2, \ldots, y_m$ y $\overline{z} \equiv z_1, z_2, \ldots, z_m$— no solo depende de los datos iniciales de entrada $x_1, x_2, \ldots, x_n$ sino que también depende de otros datos $y_1, y_2, \ldots, y_m$ y $z_1, z_2, \ldots, z_m$ que no están disponibles al principio.

En este tema se presentan dos casos concretos de cálculos que no admiten especificación funcional. Se trata de dos algoritmos que sirven para adivinar o averiguar qué fórmula lógica tiene el usuario en mente. Uno de los algoritmos es para el caso de las fórmulas k-CNF y el otro es para el caso de las fórmulas k-DNF. Estos dos tipos de fórmulas pertenecen a la lógica proposicional. Los detalles se dan en los apartados correspondientes.

También se pueden diseñar algoritmos que en vez de adivinar fórmulas de la lógica proposicional adivinan autómatas y otros conceptos matemáticos. Pero para muchos conceptos todavía no se tiene un algoritmo polinómico (es decir, eficiente).

Otro ejemplo de cálculo que no admite especificación funcional sería el del programa que juega al ajedrez contra una persona. Ese programa recibe al principio solo un dato: el color de sus fichas, es decir, blanco o negro. El programa ha de ir construyendo el resultado final,

[1] "Smart Systems" en inglés. Por consiguiente, "Inteligente" tiene aquí en parte el significado de "listo", "vivo", "hábil" y se utiliza refiriéndose a máquinas y terminales que interactúan a la vez con el usuario final y con otras fuentes de información (como bases de datos externas u otras máquinas inteligentes) a las que consulta para obtener la información que necesita para atender al usuario final.

que será el tablero con una serie de fichas en una determinada posición y cumpliéndose que el propio programa o el usuario ha perdido, o si no, cumpliéndose que se ha llegado a tablas (empate). Pero el resultado final no dependerá solo del color que le ha tocado al programa, también dependerá de los movimientos que vaya haciendo el usuario. Esos movimientos del usuario son datos que no están disponibles al principio para el programa y que le van llegando al programa durante el proceso de ejecución.

Un aspecto que siempre hay que tener presente cuando se desarrolle cualquier programa es la eficiencia (tiempo que tarda el programa en hacer sus cálculos y espacio de memoria que requiere para hacer sus cálculos). Los algoritmos que se presentarán en este tema para las fórmulas k-CNF y las fórmulas k-DNF son polinómicos, es decir, eficientes. Para otros tipos de fórmulas lógicas, se tienen algoritmos exponenciales —es decir, ineficientes— pero no se tienen algoritmos eficientes y, por lo general, no se sabe si existen algoritmos eficientes.

Recordemos que en el caso de cálculos que admiten especificación funcional, el resultado puede ser expresado utilizando los datos de entrada con los que se cuenta desde el principio. En la figura 2.1.2 de la página 14, se muestra el esquema general de un problema que admite especificación funcional. Ahí los datos de entrada son $x_1, x_2, \ldots, x_n$ y el resultado $f(x_1, x_2, \ldots, x_n)$ solo depende de los datos iniciales de entrada $x_1, x_2, \ldots, x_n$. Un ejemplo concreto sería el cálculo del factorial de un número natural x. En la tabla 2.1.1 de la página 12, se muestra una posible especificación funcional para ese caso concreto.

Especificación funcional
Descripción del dato de entrada x:
$x \in \mathbb{N}$
Descripción del resultado r:
$r = \prod_{k=1}^{x} k$

Tabla 2.1.1. Especificación funcional del problema consistente en calcular el factorial de un número natural.

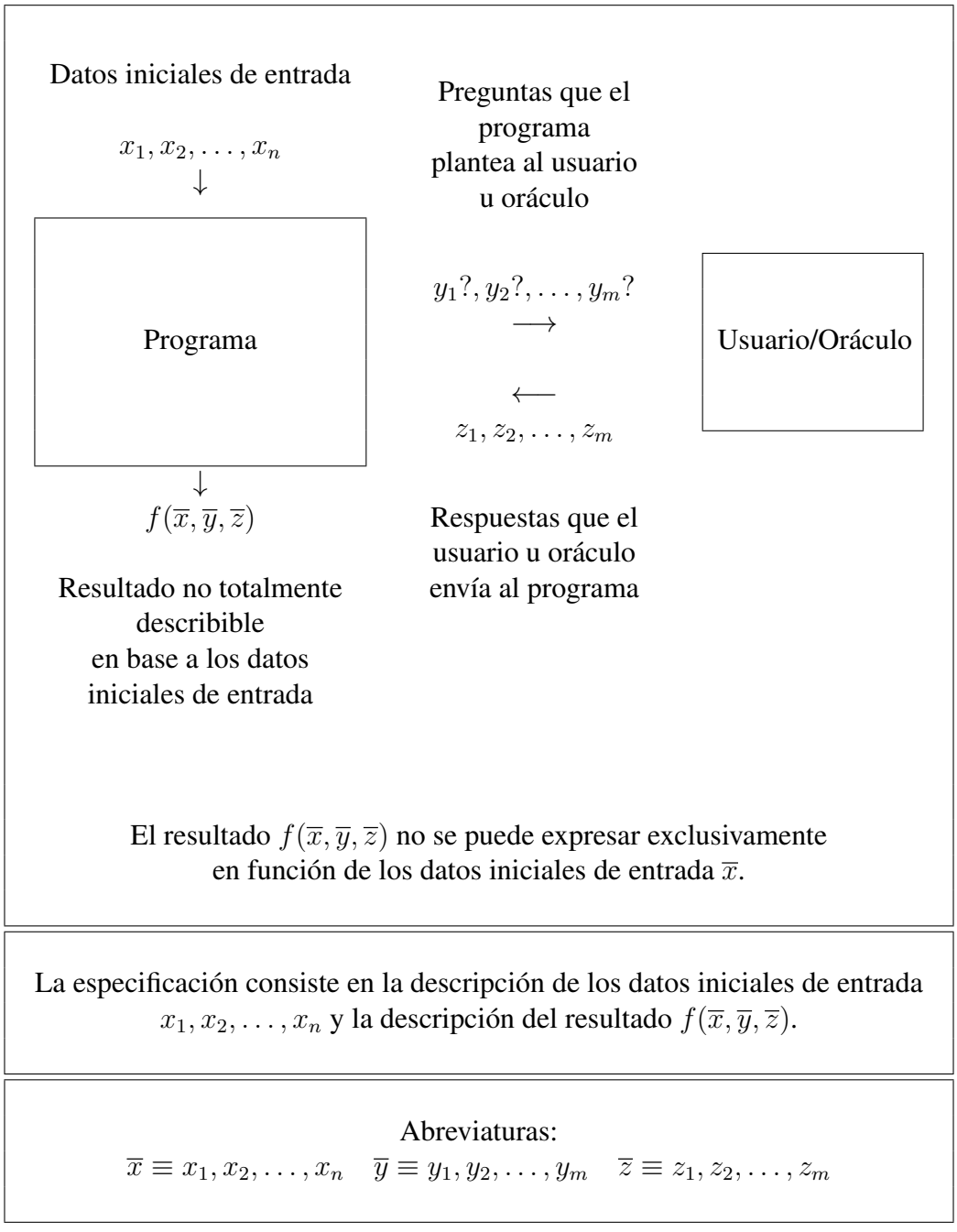

Figura 2.1.1. Esquema de especificación no funcional.

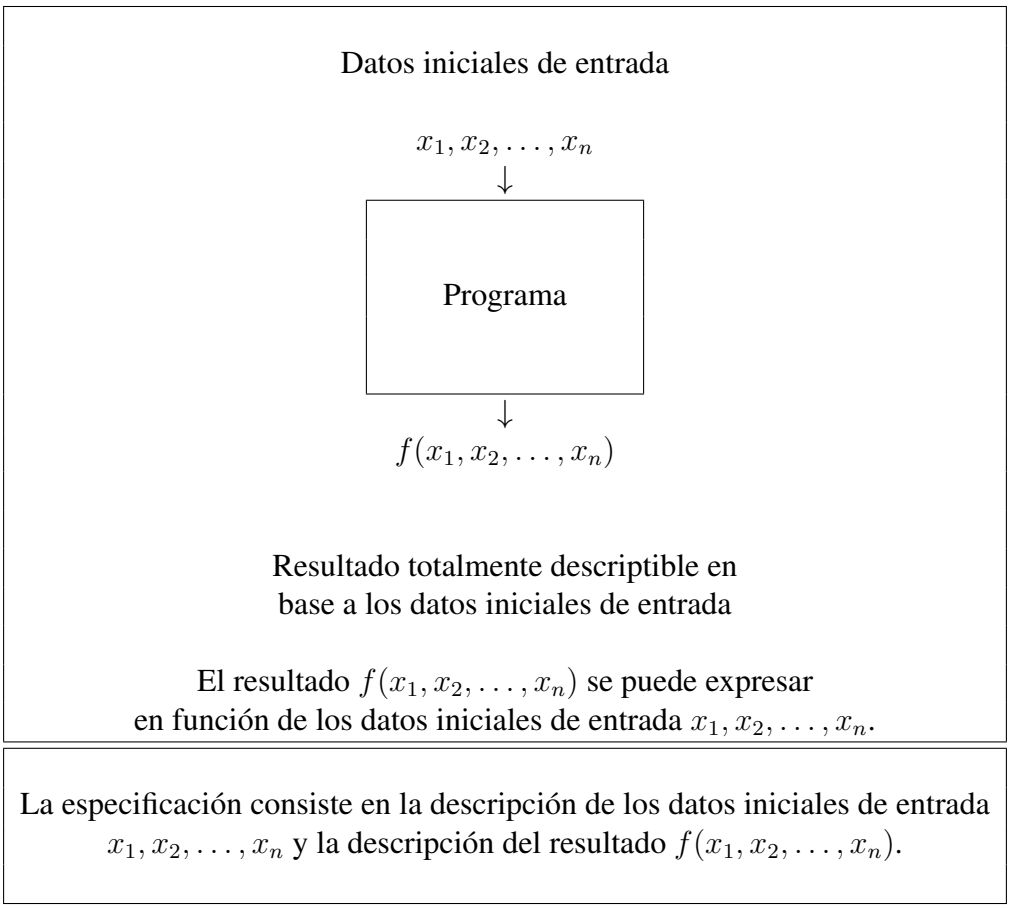

Figura 2.1.2. Esquema de especificación funcional.

2.2.

Vocabulario

En este apartado se presenta el vocabulario o la terminología relacionada con los dos tipos de fórmulas de la lógica proposicional para los cuales vamos a mostrar sendos algoritmos de adivinación.

1. **Variable proposicional**: variable cuyo valor será $True$ o $False$. Las variables proposicionales las denotaremos como x_1, x_2, x_3, etc.

2. **Literal**: variable proposicional o variable proposicional negada. Por ejemplo, x_1, x_2, x_3, $\neg x_1$, $\neg x_2$, $\neg x_3$ y $\neg x_4$ son literales. El valor de un literal será $True$ o $False$.

3. **Término**: conjunción de literales. Por ejemplo, $(\neg x_1 \wedge x_3 \wedge \neg x_8)$ es un término. El valor de un término será $True$ o $False$. Como caso particular, un término puede estar formado por un único literal. Por ejemplo, (x_4) es un término y $(\neg x_6)$ es también un término. Además, puede ocurrir que un término sea vacío, es decir, que sea la conjunción vacía, formada por cero literales. El término vacío se puede representar como $()$. El valor del término vacío es $\underline{True}$.

4. **Cláusula**: disyunción de literales. Por ejemplo, $(\neg x_2 \vee \neg x_4 \vee x_5)$ es una cláusula. El valor de una cláusula será $True$ o $False$. Como caso particular, una cláusula puede estar formada por un único literal. Por ejemplo, (x_4) es una cláusula y $(\neg x_6)$ es también una cláusula. Además, puede ocurrir que una cláusula sea vacía, es decir, que sea la disyunción vacía, formada por cero literales. La cláusula vacía se puede representar como $()$. El valor de la cláusula vacía es $\underline{False}$.

5. **k-CNF**: conjunción de cláusulas donde cada cláusula tiene como máximo k literales, donde k es un número entero no negativo, es decir, $k \geq 0$. En una k-CNF puede ocurrir que no haya ninguna cláusula que tenga exactamente k literales. Consecuentemente, una k-CNF será también una $(k+1)$-CNF. Generalizando, una k-CNF será también una $(k+j)$-CNF, donde $j \geq 0$. El valor de una k-CNF será $True$ o $False$. CNF viene del inglés Conjunctive Normal Form que en castellano sería Forma Normal Conjuntiva. Como caso particular, una k-CNF puede estar formada por una única cláusula. Además,

puede ocurrir que una k-CNF sea vacía, es decir, que sea la conjunción vacía, formada por cero cláusulas. La k-CNF vacía se representa mediante la constante $True$. A continuación, se muestran nueve ejemplos de fórmulas k-CNF:

3-CNF	$\gamma_1 \equiv$	$(\neg x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_4)$
2-CNF	$\gamma_2 \equiv$	$(\neg x_3 \vee \neg x_6) \wedge (x_2) \wedge (x_3 \vee x_7) \wedge (x_8) \wedge (\neg x_5 \vee \neg x_{12})$
2-CNF	$\gamma_3 \equiv$	$(\neg x_3 \vee \neg x_6) \wedge (x_3 \vee x_7) \wedge (\neg x_5 \vee \neg x_{12})$
1-CNF	$\gamma_4 \equiv$	$(\neg x_4) \wedge (x_2) \wedge (\neg x_{12})$
3-CNF	$\gamma_5 \equiv$	$(x_1 \vee \neg x_2 \vee x_4)$
2-CNF	$\gamma_6 \equiv$	$(\neg x_2 \vee \neg x_{10})$
1-CNF	$\gamma_7 \equiv$	$(\neg x_8)$
4-CNF	$\gamma_8 \equiv$	$(\neg x_3 \vee \neg x_6 \vee x_8 \vee x_9) \wedge (x_7) \wedge (x_2 \vee \neg x_4 \vee x_5 \vee x_{11})$
1-CNF	$\gamma_9 \equiv$	$(x_1) \wedge (\neg x_2) \wedge (x_4)$
2-CNFa	$\gamma_{10} \equiv$	$(\neg x_3 \vee \neg x_6) \wedge () \wedge (x_3 \vee x_7) \wedge (\neg x_5 \vee \neg x_{12})$
0-CNFa	$\gamma_{11} \equiv$	$()$
0-CNFa	$\gamma_{12} \equiv$	$True$

Nótese que γ_1 es una 4-CNF, es una 5-CNF, es una 6-CNF, etcétera. En general, es una $(3+j)$-CNF donde $j \geq 0$. De la misma forma, γ_4 es una $(1+j)$-CNF donde $j \geq 0$. Y γ_8 es una $(4+j)$-CNF donde $j \geq 0$. Por otro lado, en las fórmulas γ_{10} y γ_{11} aparece la cláusula vacía: $()$. El valor de esas dos fórmulas es $False$. La fórmula γ_{11} es una 0-CNF porque sus cláusulas contienen como máximo 0 literales. Pero al mismo tiempo, γ_{11} es una $(0+j)$-CNF, para cualquier $j \geq 0$. La fórmula γ_{12} es la k-CNF vacía, no tiene ninguna cláusula. También γ_{12} es una 0-CNF porque en sus cláusulas contienen como máximo 0 literales. Pero al mismo tiempo, γ_{12} es una $(0+j)$-CNF, para cualquier $j \geq 0$.

6. **k-DNF**: disyunción de términos donde cada término tiene como máximo k literales, donde k es un número entero no negativo, es decir, $k \geq 0$. En una k-DNF puede ocurrir que no haya ningún término que tenga exactamente k literales. Consecuentemente, una k-DNF será también una $(k+1)$-DNF. Generalizando, una k-DNF será también una $(k+j)$-DNF, donde $j \geq 0$. El valor de una k-DNF será $True$ o $False$. DNF viene del inglés Disjunctive Normal Form que en castellano sería Forma Normal Disyuntiva. Como caso particular, una k-DNF puede estar formada por un único término. Además, puede ocurrir que una k-DNF sea vacía, es decir, que sea la disyunción vacía, formada por cero términos. La k-DNF vacía se representa mediante la constante $False$. A continuación, se muestran nueve ejemplos de fórmulas k-DNF:

3-DNF	$\delta_1 \equiv$	$(\neg x_3) \vee (x_1 \wedge \neg x_2 \wedge x_4) \vee (x_2 \wedge x_3) \vee (\neg x_1 \wedge \neg x_4)$
2-DNF	$\delta_2 \equiv$	$(\neg x_3 \wedge \neg x_6) \vee (x_2) \vee (x_3 \wedge x_7) \vee (x_8) \vee (\neg x_5 \wedge \neg x_{12})$
2-DNF	$\delta_3 \equiv$	$(\neg x_3 \wedge \neg x_6) \vee (x_3 \wedge x_7) \vee (\neg x_5 \wedge \neg x_{12})$
1-DNF	$\delta_4 \equiv$	$(\neg x_4) \vee (x_2) \vee (\neg x_{12})$
3-DNF	$\delta_5 \equiv$	$(x_1 \wedge \neg x_2 \wedge x_4)$
2-DNF	$\delta_6 \equiv$	$(\neg x_2 \wedge \neg x_{10})$
1-DNF	$\delta_7 \equiv$	$(\neg x_8)$
4-DNF	$\delta_8 \equiv$	$(\neg x_3 \wedge \neg x_6 \wedge x_8 \wedge x_9) \vee (x_7) \vee (x_2 \wedge \neg x_4 \wedge x_5 \wedge x_{11})$
1-DNF	$\delta_9 \equiv$	$(x_1) \vee (\neg x_2) \vee (x_4)$
2-DNFa	$\delta_{10} \equiv$	$(\neg x_3 \wedge \neg x_6) \vee () \vee (x_3 \wedge x_7) \vee (\neg x_5 \wedge \neg x_{12})$
0-DNFa	$\delta_{11} \equiv$	$()$
0-DNFa	$\delta_{12} \equiv$	$False$

Nótese que δ_1 es una 4-DNF, es una 5-DNF, es una 6-DNF, etcétera. En general, es una $(3+j)$-DNF donde $j \geq 0$. De la misma forma, δ_4 es una $(1+j)$-DNF donde $j \geq 0$. Y δ_8 es una $(4+j)$-DNF donde $j \geq 0$. Por otro lado, en las fórmulas δ_{10} y δ_{11} aparece el término vacío: $()$. El valor de esas dos fórmulas es $True$. La fórmula δ_{11} es una 0-DNF porque sus términos contienen como máximo 0 literales. Pero al mismo tiempo, δ_{11} es una $(0+j)$-DNF, para cualquier $j \geq 0$. La fórmula δ_{12} es la k-DNF vacía, no tiene ningún término. También δ_{12} es una 0-DNF porque en sus términos contienen como máximo 0 literales. Pero al mismo tiempo, δ_{12} es una $(0+j)$-DNF, para cualquier $j \geq 0$.

Por otro lado, nótese también que γ_5 y δ_9 son la misma fórmula lógica. Pero esa fórmula lógica puede ser interpretada como 3-CNF formada por una única cláusula (γ_5) o como una 1-DNF formada por tres términos (δ_9). De la misma forma, γ_9 y δ_5 son la misma fórmula lógica. Pero esa fórmula lógica puede ser interpretada como una 1-CNF formada por tres cláusulas (γ_9) o como una 3-DNF formada por un único término (δ_5). Por otra parte, γ_7 y δ_7 son la misma fórmula. Esa fórmula admite ser interpretada como una 1-CNF formada por una única cláusula de un único literal (γ_7) o como una 1-DNF formada por un único término de un único literal (δ_7).

7. **Valoración**: dada una fórmula lógica γ definida sobre unas variables $x_1, x_2, \ldots, x_n$, una valoración para γ es una tupla ordenada de n elementos donde cada elemento es un $True$ o un $False$. Una valoración sirve para asignar el valor $True$ o $False$ a cada variable proposicional y lo representaremos de la siguiente forma:

   $$(\underbrace{\beta_1}_{x_1}, \underbrace{\beta_2}_{x_2}, \ldots, \underbrace{\beta_n}_{x_n}) \text{ donde cada para cada } i \in \{1, 2, \ldots, n\},$$

   se tiene que $\beta_i \in \{True, False\}$

   En las valoraciones se ha de respetar el orden de las variables teniendo en cuenta el subíndice: de 1 a n. De ahí que una valoración sea una tupla ordenada.

Por ejemplo, $v_1 = (True, True, True)$ y $v_2 = (True, False, False)$ son dos valoraciones para la fórmula lógica $\gamma = (x_1 \vee x_2) \wedge (x_2 \vee x_3)$. Puesto que γ está definida sobre tres variables, las valoraciones tienen tres componentes para así asignar un valor a cada variable en el orden (x_1, x_2, x_3). Para la valoración v_1 la fórmula γ es $True$ mientras que para la valoración v_2 la fórmula γ es $False$. Para tres variables se pueden generar 8 valoraciones distintas: 2^3. En general, para n variables, habrá 2^n valoraciones distintas.

Aunque se haya establecido un conjunto de variables proposicionales $x_1, x_2, \ldots, x_n$ para ser utilizadas en las fórmulas, no es necesario que todas las variables aparezcan en todas las fórmulas. Por ejemplo, si se ha establecido que las variables a utilizar son x_1, x_2, x_3 y x_4, se considera que la fórmula $\delta = (\neg x_2) \wedge (x_3 \vee x_4)$ es una fórmula definida sobre x_1, x_2, x_3 y x_4, aunque x_1 no aparezca. Pero las valoraciones han de ser de tamaño cuatro. Consecuentemente, $(\underbrace{True}_{x_1}, \underbrace{False}_{x_2}, \underbrace{False}_{x_3}, \underbrace{True}_{x_4})$ es una valoración correcta para δ. Lo que ocurrirá es que el valor de x_1 no influirá en el valor de δ porque x_1 no aparece en δ.

8. **Fórmulas equivalentes**: dadas dos fórmulas proposicionales γ y δ definidas sobre unas mismas variables $x_1, x_2, \ldots, x_n$, se dice que γ y δ son equivalentes si para todas las valoraciones de tamaño n, el valor de γ y el valor de δ coinciden.

   A continuación se muestran cuatro ejemplos de fórmulas equivalentes y un caso de fórmulas no equivalentes. Todas las fórmulas estan definidas sobre las variables x_1, x_2, x_3 y x_4:

   - $\gamma_1 = \neg(x_1 \wedge x_2 \wedge x_3)$ y $\delta_1 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$ son fórmulas equivalentes.
   - $\gamma_2 = (x_1 \vee x_2) \wedge (x_2) \wedge (x_3)$ y $\delta_2 = (x_2) \wedge (x_3)$ son fórmulas equivalentes.
   - $\gamma_3 = (x_1 \vee x_2) \wedge (x_4)$ y $\delta_3 = (x_1 \wedge x_4) \vee (x_2 \wedge x_4)$ son fórmulas equivalentes.
   - $\gamma_4 = (\neg x_1 \vee \neg x_3) \wedge (x_3)$ y la siguiente fórmula δ_4 son equivalentes:

     $$\begin{aligned}\delta_4 = \quad & (\neg x_1) \wedge (x_3) \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3)\end{aligned}$$

     Para darnos cuenta de que γ_4 y δ_4 son equivalentes, conviene percatarse de lo siguiente:

     * $\gamma_4 = (\neg x_1 \vee \neg x_3) \wedge (x_3)$ es equivalente a $\gamma_4' = (\neg x_1) \wedge (x_3)$
     * En δ_4 tenemos, por una parte, que $(\neg x_1) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$ es equivalente a $(\neg x_1)$. Por otra parte, $(x_3) \wedge (x_1 \vee x_3) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3)$ es equivalente a (x_3). En definitiva, δ_4 es equivalente a $\delta_4' = (\neg x_1) \wedge (x_3)$.
     * Por tanto, γ_4' y δ_4' son la misma fórmula, y esto quiere decir que γ_4 y δ_4 son equivalentes.

   - $\gamma_5 = (x_1 \vee x_2) \wedge (x_2 \vee x_3)$ y $\delta_5 = (x_1 \vee x_2) \wedge x_3$ **no son equivalentes** ya que para la valoración $v = (False, True, False)$ la fórmula γ_5 es $True$ y la fórmula δ_5 es $False$.

2.3.

Algoritmo de aprendizaje o adivinación para las k-CNF

En este apartado se presenta un algoritmo que es capaz de adivinar o aprender una fórmula proposicional k-CNF a base de hacer preguntas o de pedir pistas al usuario.

El proceso llevado a cabo por el algoritmo puede ser interpretado como adivinación o como aprendizaje de una fórmula:

- Adivinación porque a partir de unas pistas —valoraciones que hacen que la fórmula que el usuario tiene en mente sea cierta— obtiene la fórmula proposicional. Es decir, adivina o averigua cuál es la fórmula.
- Aprendizaje porque a partir de unos ejemplos —valoraciones que hacen que la fórmula que el usuario tiene en mente sea cierta— obtiene la regla general. Es decir, aprende el funcionamiento general de la fórmula.

2.3.1 Escenario

1. El usuario tiene en mente una k-CNF (a la que llamaremos g) en el que pueden aparecer las variables $x_1, x_2, \ldots, x_n$. Pero no es necesario que todas esas variables aparezcan. Además, tiene que cumplirse $n \geq 0$ y $0 \leq k \leq n$.

   Por ejemplo, si k es 2 y n es 4 el usuario podría tener en mente la siguiente fórmula 2-CNF:

   $$g = (\neg x_3) \wedge (x_2 \vee x_3) \wedge (x_4)$$

   en la cual la variable proposicional x_1 no aparece.

2. El programa pide al usuario los valores k y n. Estos son los dos únicos datos iniciales que recibe el algoritmo.

3. El algoritmo empieza con el proceso de adivinación:

3.1. El algoritmo genera la k-CNF inicial más general posible utilizando las variables $x_1, x_2, \ldots, x_n$. Llamaremos h_0 a esa k-CNF inicial. Las cláusulas que conforman h_0 tendrán como máximo, tamaño k.

Por ejemplo, si $k = 2$ y $n = 2$, la fórmula más general posible es la siguiente 2-CNF:

$$h_0 = ()\wedge(x_1)\wedge(\neg x_1)\wedge(x_2)\wedge(\neg x_2)\wedge(x_1\vee x_2)\wedge(x_1\vee\neg x_2)\wedge(\neg x_1\vee x_2)\wedge(\neg x_1\vee\neg x_2)$$

Es decir, una fórmula en la que aparecen x_1 y x_2 (por ser $n = 2$) y que es la conjunción de todas las cláusulas relevantes de cero literales, un literal y dos literales, por ser $k = 2$.

Por equivalencias lógicas, no habría que poner, por ejemplo, la cláusula $(x_2 \vee x_1)$ porque es equivalente a $(x_1 \vee x_2)$ y, por tanto, no es relevante, no aporta nada. Tampoco habría que poner, por ejemplo, la cláusula $(x_1 \vee x_1)$ porque es equivalente a (x_1) y no aportaría nada. Y tampoco habría que poner, por ejemplo, la cláusula $(x_1 \vee \neg x_1)$ porque es equivalente a $True$ y puesto que $\delta \wedge True$ es equivalente a δ la cláusula $(x_1 \vee \neg x_1)$ no aporta nada.

Si n fuera 3 y k fuera 2, habría que utilizar las variables x_1, x_2 y x_3, por ser $n = 3$, y habría que generar todas las cláusulas relevantes de cero literales, un literal y dos literales, por ser $k = 2$:

$$\begin{aligned} h_0 = \; & () \wedge \\ & (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge \\ & (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

Si n fuera 5 y k fuera 1, habría que utilizar las variables x_1, x_2, x_3, x_4 y x_5, por ser $n = 5$, y habría que generar todas las cláusulas relevantes de cero literales y un literal, por ser $k = 1$:

$$h_0 = () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge (x_4) \wedge (\neg x_4) \wedge (x_5) \wedge (\neg x_5)$$

3.2. El algoritmo presenta h_0 al usuario y le pregunta si h_0 es equivalente a g.

3.3. El usuario responde con un "Sí" o con un "No".

3.3.1. Si la respuesta del usuario ha sido "Sí", ello quiere decir que las fórmulas h_0 y g son equivalentes y termina la adivinación.

3.3.2. Si la respuesta del usuario ha sido "No", ello quiere decir que h_0 y g no son equivalentes y, consecuentemente, el algoritmo pide al usuario una valoración

v_1 que haga cierta a g y falsa a h_0. Utilizando v_1, el algoritmo elimina de h_0 todas las cláusulas que son falsas con v_1 obteniendo así una nueva propuesta h_1, y vuelve al punto 3.2 donde realiza con h_1 lo indicado ahí para h_0. Se repite el proceso hasta dar con una h_j que sea equivalente a g.

2.3.2 Algoritmo para las k-CNF

En la tabla 2.3.1 de la página 22, se muestra el algoritmo correspondiente al problema de adivinar una k-CNF g que el usuario tiene en mente. Puesto que el algoritmo va obteniendo datos desde el exterior a lo largo del proceso de cálculo, decimos que es un sistema inteligente.

2.3.3 Tipos de preguntas o peticiones al usuario: k-CNF

El algoritmo de las k-CNF realiza dos tipos de preguntas al usuario:

- Pedir una valoración v_{j+1} que hace $True$ a g y $False$ a h_j.
- Preguntar al usuario si h_j es equivalente a g.

2.3.4 Especificación no funcional: k-CNF

En la tabla 2.3.2 de la página 23, se muestra la especificación no funcional correspondiente al problema de adivinar una k-CNF g que el usuario tiene en mente.

Los datos iniciales de entrada para el algoritmo serán n y k. El resultado será una k-CNF h. Pero esa k-CNF h no puede ser descrita utilizando solo los datos iniciales n y k. Dicho de otra forma, la k-CNF h no puede ser descrita en función de los datos iniciales n y k. Debido a ello, la especificación es no funcional.

Para adivinar la k-CNF, el algoritmo irá presentando sus hipótesis h_j al usuario. Cada vez que la hipótesis h_j presentada no sea equivalente a g, el usuario dará una pista v_{j+1}, es decir, una valoración que hace falsa a h_j pero cierta a g. Como consecuencia de esto, el resultado final h depende también de g y de las pistas o valoraciones que el usuario dé durante el proceso, pero ni g ni las pistas están disponibles para el algoritmo desde el principio.

{Precondición para datos iniciales de entrada:
$n \in \mathbb{N} \wedge n \geq 0 \wedge k \in \mathbb{N} \wedge k \geq 0 \wedge k \leq n$}

{Precondición para la fórmula g (esta fórmula no es conocida por el algoritmo):
$es\_k\_CNF(g)$}

$j := 0$;
$h_j := k\_CNF\_inicial(n, k)$;
$equiv := preguntar\_al\_usuario\_si\_la\_formula\_es\_equivalente(h_j)$;

mientras $\neg equiv$ **hacer**

$v_{j+1} := pedir\_al\_usuario\_valoracion\_que\_hace\_falsa\_a(h_j)$;
$h_{j+1} := eliminar\_clausulas\_falsas(h_j, v_{j+1})$;
$j := j + 1$;
$equiv := preguntar\_al\_usuario\_si\_la\_formula\_es\_equivalente(h_j)$;

fin mientras

$h := h_j$;

{Postcondición: $es\_k\_CNF(h) \wedge (h \leftrightarrow g)$}

Tabla 2.3.1. Algoritmo que sirve para aprender o adivinar una fórmula k-CNF g.

Especificación no funcional para el problema de adivinar las k-CNF
Descripción de los datos de entrada iniciales n y k:
$n \in \mathbb{N} \wedge n \geq 0 \wedge k \in \mathbb{N} \wedge k \geq 0 \wedge k \leq n$ donde n indica que se pueden utilizar n variables proposicionales $x_1, x_2, \ldots, x_n$ y k indica que las cláusulas que conforman la k-CNF pueden tener, como máximo k literales.
Descripción del resultado h:
h es una fórmula k-CNF y se cumple $h \leftrightarrow g$ (es decir, h es equivalente a g) donde g es la fórmula k-CNF que el usuario tiene en mente.

Tabla 2.3.2. Especificación no funcional del problema consistente en aprender o adivinar una fórmula k-CNF g, que el usuario tiene en mente.

2.4.

Ejercicios resueltos para las k-CNF

En este apartado se presentan algunos ejercicios de examen resueltos para el algoritmo de las k-CNF. Por tanto, lo que se hace en estos ejercicios resueltos es lo que hay que hacer en el examen.

El proceso de adivinación se desarrolla como un diálogo entre el algoritmo (A) y el usuario (U). El usuario tiene en mente una fórmula k-CNF g y el algoritmo en ningún momento dispone de esa k-CNF g. El algoritmo presenta su propuesta h_j al usuario y si h_j no es equivalente a g, el usuario le da una pista: una valoración v_{j+1} que hace cierta a g y falsa a h_j. Para tener una propuesta que se parezca más a g, el algoritmo elimina de h_j aquellas cláusulas que son falsas con v_{j+1} y así construye una nueva propuesta h_{j+1} que se parece más a g porque h_{j+1} será cierta con v_{j+1}. El proceso se repite hasta que el algoritmo llega a tener una propuesta que es equivalente a g.

Al hacer el ejercicio, nuestro papel es el del algoritmo. Tenemos que ir construyendo propuestas h_j y preguntando al usuario si h_j es equivalente a g y, en caso de que no sea equivalente, tenemos que pedir una valoración v_{j+1} que hace cierta a g y falsa a h_j. Pero al desarrollar este diálogo, el usuario no estará. Debido a ello, en el propio enunciado se nos dan las pistas que el usuario debería darnos cada vez que solicitamos una pista. Hay que agotar y utilizar todas las pistas. Esto significa que cada vez que presentamos una propuesta h_j al usuario, si todavía hay pistas no utilizadas, tenemos que suponer que el usuario nos dirá que no es equivalente a g y que, a continuación, tenemos que utilizar la siguiente pista que todavía no ha sido utilizada. La k-CNF que tengamos cuando se hayan agotado todas las pistas, será la k-CNF definitiva.

Teniendo en cuenta lo que se acaba de indicar, en el ejemplo 1 que viene a continuación, habrá que construir una propuesta inicial h_0 y cuatro propuestas h_1, h_2, h_3 y h_4, cada una de las cuales se obtendrá a partir de la propuesta anterior y utilizando la valoración correspondiente. Por ejemplo, h_3 se construirá a partir de h_2 y v_3. Al haber cuatro valoraciones —v_1, v_2, v_3 y v_4— hay que llegar hasta h_4.

2.4.1 Ejemplo 1 de k-CNF: $k = 2$ y $n = 3$

2.4.1.1 Enunciado

El usuario tiene en mente una fórmula 2-CNF g que puede hacer uso de 3 variables x_1, x_2 y x_3. Por tanto, tenemos que $k = 2$ y $n = 3$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, T, F)$
- $v_2 = (T, F, T)$
- $v_3 = (T, F, F)$
- $v_4 = (F, T, F)$

2.4.1.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 3$

A: Propuesta inicial:

$$\begin{aligned} h_0 \;=\; & () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, T, F)$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \;=\; & \cancel{()} \wedge (x_1) \wedge \cancel{(\neg x_1)} \wedge (x_2) \wedge \cancel{(\neg x_2)} \wedge \cancel{(x_3)} \wedge (\neg x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge \cancel{(\neg x_1 \vee \neg x_2)} \wedge \cancel{(\neg x_1 \vee x_3)} \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge \cancel{(\neg x_2 \vee x_3)} \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_1 \quad = \quad & (x_1) \ \wedge \ (x_2) \ \wedge \ (\neg x_3) \ \wedge \\ & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \ \wedge \ (x_1 \vee \neg x_3) \ \wedge \\ & (\neg x_1 \vee x_2) \ \wedge \ (\neg x_1 \vee \neg x_3) \ \wedge \\ & (x_2 \vee x_3) \ \wedge \ (x_2 \vee \neg x_3) \ \wedge \ (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (T, F, T)$ hace que g sea $True$ y h_1 sea $False$.

A: (Se eliminan las cláusulas de h_1 que son $False$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \quad = \quad & (x_1) \ \wedge \ \cancel{(x_2)} \ \wedge \ \cancel{(\neg x_3)} \ \wedge \\ & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \ \wedge \ (x_1 \vee \neg x_3) \ \wedge \\ & \cancel{(\neg x_1 \vee x_2)} \ \wedge \ \cancel{(\neg x_1 \vee \neg x_3)} \ \wedge \\ & (x_2 \vee x_3) \ \wedge \ \cancel{(x_2 \vee \neg x_3)} \ \wedge \ (\neg x_2 \vee \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_2 \quad = \quad & (x_1) \ \wedge \\ & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \ \wedge \ (x_1 \vee \neg x_3) \ \wedge \\ & (x_2 \vee x_3) \ \wedge \ (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (T, F, F)$ hace que g sea $True$ y h_2 sea $False$.

A: (Se eliminan las cláusulas de h_2 que son $False$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{aligned} h_2 \quad = \quad & (x_1) \ \wedge \\ & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \ \wedge \ (x_1 \vee \neg x_3) \ \wedge \\ & \cancel{(x_2 \vee x_3)} \ \wedge \ (\neg x_2 \vee \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_3 \quad = \quad & (x_1) \ \wedge \\ & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \ \wedge \ (x_1 \vee \neg x_3) \ \wedge \\ & (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (F, T, F)$ hace que g sea $True$ y h_3 sea $False$.

A: (Se eliminan las cláusulas de h_3 que son $False$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$\begin{array}{rcl} h_3 & = & \cancel{(x_1)} \wedge \\ & & (x_1 \vee x_2) \ \wedge \ \cancel{(x_1 \vee \neg x_2)} \ \wedge \ \cancel{(x_1 \vee x_3)} \ \wedge \ (x_1 \vee \neg x_3) \wedge \\ & & (\neg x_2 \vee \neg x_3) \end{array}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{array}{rcl} h_4 & = & (x_1 \vee x_2) \ \wedge \ (x_1 \vee \neg x_3) \wedge \\ & & (\neg x_2 \vee \neg x_3) \end{array}$$

¿$h_4 \leftrightarrow g$?

U: Sí.

2.4.1.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $() \wedge (x_1) \wedge (\neg x_1) \wedge \ldots$. El valor de la cláusula vacía $()$ es $False$ y, por equivalencias de la lógica, tenemos que $False \wedge \gamma \equiv False$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \wedge (\neg x_1)$ y se cumple que $\delta \wedge \neg\delta \equiv False$ para cualquier fórmula lógica δ. Por tanto, h_0 es $False$ por más de un motivo.

2.4.2 Ejemplo 2 de k-CNF: $k = 2$ y $n = 2$ (simplificable)

2.4.2.1 Enunciado

El usuario tiene en mente una fórmula 2-CNF g que puede hacer uso de 2 variables x_1 y x_2. Por tanto, tenemos que $k = 2$ y $n = 2$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, F)$
- $v_2 = (F, F)$

2.4.2.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 2$.

A: Propuesta inicial:

$$\begin{aligned} h_0 \quad = \quad & () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, F)$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \quad = \quad & \cancel{()} \wedge (x_1) \wedge \cancel{(\neg x_1)} \wedge \cancel{(x_2)} \wedge (\neg x_2) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \\ & \cancel{(\neg x_1 \vee x_2)} \wedge (\neg x_1 \vee \neg x_2) \end{aligned}$$

Por lo tanto, la nueva propuesta es la siguiente:

$$\begin{aligned} h_1 \quad = \quad & (x_1) \wedge (\neg x_2) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge \\ & (\neg x_1 \vee \neg x_2) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (F, F)$ hace que g sea $True$ y h_1 sea $False$.

A: (Se eliminan las cláusulas de h_1 que son $False$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \quad = \quad & \cancel{(x_1)} \wedge (\neg x_2) \wedge \\ & \cancel{(x_1 \vee x_2)} \wedge (x_1 \vee \neg x_2) \wedge \\ & (\neg x_1 \vee \neg x_2) \end{aligned}$$

Por lo tanto, la nueva propuesta es la siguiente:

$$\begin{aligned} h_2 \quad = \quad & (\neg x_2) \ \wedge \\ & (x_1 \vee \neg x_2) \ \wedge \\ & (\neg x_1 \vee \neg x_2) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: Sí.

2.4.2.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $() \wedge (x_1) \wedge (\neg x_1) \wedge \ldots$. El valor de la cláusula vacía $()$ es $False$ y, por equivalencias de la lógica, tenemos que $False \wedge \gamma \equiv False$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \wedge (\neg x_1)$ y se cumple que $\delta \wedge \neg\delta \equiv False$ para cualquier fórmula lógica δ. Por tanto, h_0 es $False$ por más de un motivo.

La k-CNF h_2 que se ha obtenido al final es equivalente a $(\neg x_2)$. Por tanto, el algoritmo no obtiene la versión más sencilla de la fórmula pero eso no importa. Nos tenemos que quedar con la fórmula que ha obtenido el algoritmo y no tenemos que preocuparnos de simplificarla.

2.4.3 Ejemplo 3 de k-CNF: $k = 2$ y $n = 3$ (simplificable)

2.4.3.1 Enunciado

El usuario tiene en mente una fórmula 2-CNF g que puede hacer uso de 3 variables x_1, x_2 y x_3. Por tanto, tenemos que $k = 2$ y $n = 3$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (F, T, T)$
- $v_2 = (F, F, T)$

2.4.3.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 3$

A: Propuesta inicial:

$$\begin{aligned} h_0 = \; & () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (F, T, T)$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 = \; & \cancel{()} \wedge \cancel{(x_1)} \wedge (\neg x_1) \wedge (x_2) \wedge \cancel{(\neg x_2)} \wedge (x_3) \wedge \cancel{(\neg x_3)} \wedge \\ & (x_1 \vee x_2) \wedge \cancel{(x_1 \vee \neg x_2)} \wedge (x_1 \vee x_3) \wedge \cancel{(x_1 \vee \neg x_3)} \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge \cancel{(\neg x_2 \vee \neg x_3)} \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_1 = \; & (\neg x_1) \wedge (x_2) \wedge (x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (F, F, T)$ hace que g sea $True$ y h_1 sea $False$.

A: (Se eliminan las cláusulas de h_1 que son $False$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 = \; & (\neg x_1) \wedge \cancel{(x_2)} \wedge (x_3) \wedge \\ & \cancel{(x_1 \vee x_2)} \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge \cancel{(x_2 \vee \neg x_3)} \wedge (\neg x_2 \vee x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$h_2 = \begin{array}{l}(\neg x_1) \wedge (x_3) \wedge \\ (x_1 \vee x_3) \wedge \\ (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3)\end{array}$$

¿$h_2 \leftrightarrow g$?

U: Sí.

2.4.3.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $() \wedge (x_1) \wedge (\neg x_1) \wedge \ldots$. El valor de la cláusula vacía $()$ es $False$ y, por equivalencias de la lógica, tenemos que $False \wedge \gamma \equiv False$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \wedge (\neg x_1)$ y se cumple que $\delta \wedge \neg\delta \equiv False$ para cualquier fórmula lógica δ. Por tanto, h_0 es $False$ por más de un motivo.

La k-CNF h_2 que se ha obtenido al final es equivalente a $(\neg x_1) \wedge (x_3)$. Por tanto, el algoritmo no obtiene la versión más sencilla de la fórmula pero eso no importa. Nos tenemos que quedar con la fórmula que ha obtenido el algoritmo y no tenemos que preocuparnos de simplificarla.

2.4.4 Ejemplo 4 de k-CNF: $k = 1$ y $n = 6$

2.4.4.1 Enunciado

El usuario tiene en mente una fórmula 1-CNF g que puede hacer uso de 6 variables x_1, x_2, x_3, x_4, x_5 y x_6. Por tanto, tenemos que $k = 1$ y $n = 6$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, F, T, T, F, T)$
- $v_2 = (T, F, T, T, F, F)$
- $v_3 = (T, T, F, T, F, T)$
- $v_4 = (T, T, F, F, F, F)$

2.4.4.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 1$ y $n = 6$

A: Propuesta inicial:

$$\begin{aligned} h_0 \quad = \quad & () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge \\ & (x_4) \wedge (\neg x_4) \wedge (x_5) \wedge (\neg x_5) \wedge (x_6) \wedge (\neg x_6) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, F, T, T, F, T)$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \quad = \quad & \cancel{()} \wedge (x_1) \wedge \cancel{(\neg x_1)} \wedge \cancel{(x_2)} \wedge (\neg x_2) \wedge (x_3) \wedge \cancel{(\neg x_3)} \wedge \\ & (x_4) \wedge \cancel{(\neg x_4)} \wedge \cancel{(x_5)} \wedge (\neg x_5) \wedge (x_6) \wedge \cancel{(\neg x_6)} \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_1 \quad = \quad & (x_1) \wedge (\neg x_2) \wedge (x_3) \wedge \\ & (x_4) \wedge (\neg x_5) \wedge (x_6) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (T, F, T, T, F, F)$ hace que g sea $True$ y h_1 sea $False$.

A: (Se eliminan las cláusulas de h_1 que son $False$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \quad = \quad & (x_1) \wedge (\neg x_2) \wedge (x_3) \wedge \\ & (x_4) \wedge (\neg x_5) \wedge \cancel{(x_6)} \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_2 \quad = \quad & (x_1) \wedge (\neg x_2) \wedge (x_3) \wedge \\ & (x_4) \wedge (\neg x_5) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (T, T, F, T, F, T)$ hace que g sea $True$ y h_2 sea $False$.

A: (Se eliminan las cláusulas de h_2 que son $False$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{array}{rcl} h_2 & = & (x_1) \wedge \cancel{(\neg x_2)} \wedge \cancel{(x_3)} \wedge \\ & & (x_4) \wedge (\neg x_5) \end{array}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{array}{rcl} h_3 & = & (x_1) \wedge \\ & & (x_4) \wedge (\neg x_5) \end{array}$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (T, T, F, F, F, F)$ hace que g sea $True$ y h_3 sea $False$.

A: (Se eliminan las cláusulas de h_3 que son $False$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$\begin{array}{rcl} h_3 & = & (x_1) \wedge \\ & & \cancel{(x_4)} \wedge (\neg x_5) \end{array}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{array}{rcl} h_4 & = & (x_1) \wedge \\ & & (\neg x_5) \end{array}$$

¿$h_4 \leftrightarrow g$?

U: Sí.

2.4.4.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $() \wedge (x_1) \wedge (\neg x_1) \wedge \ldots$. El valor de la cláusula vacía $()$ es $False$ y, por equivalencias de la lógica, tenemos que $False \wedge \gamma \equiv False$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \wedge (\neg x_1)$ y se cumple que $\delta \wedge \neg\delta \equiv False$ para cualquier fórmula lógica δ. Por tanto, h_0 es $False$ por más de un motivo.

2.4.5 Ejemplo 5 de k-CNF: $k = 2$ y $n = 3$ (True – Conjunción vacía)

2.4.5.1 Enunciado

El usuario tiene en mente una fórmula 2-CNF g que puede hacer uso de 3 variables x_1, x_2 y x_3. Por tanto, tenemos que $k = 2$ y $n = 3$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (F, F, T)$
- $v_2 = (T, F, T)$
- $v_3 = (T, T, T)$
- $v_4 = (F, F, F)$
- $v_5 = (F, T, F)$
- $v_6 = (T, F, F)$

2.4.5.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 3$

A: Propuesta inicial:

$$\begin{aligned} h_0 \;=\; & () \wedge (x_1) \wedge (\neg x_1) \wedge (x_2) \wedge (\neg x_2) \wedge (x_3) \wedge (\neg x_3) \wedge \\ & (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee \neg x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (F, F, T)$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \;=\; & \cancel{()} \wedge \cancel{(x_1)} \wedge (\neg x_1) \wedge \cancel{(x_2)} \wedge (\neg x_2) \wedge (x_3) \wedge \cancel{(\neg x_3)} \wedge \\ & \cancel{(x_1 \vee x_2)} \wedge (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge \cancel{(x_1 \vee \neg x_3)} \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge \cancel{(x_2 \vee \neg x_3)} \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_1 \;=\; & (\neg x_1) \wedge (\neg x_2) \wedge (x_3) \wedge \\ & (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3) \wedge \\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (T, F, T)$ hace que g sea $True$ y h_1 sea $False$.

A: (Se eliminan las cláusulas de h_1 que son $False$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \;=\; & \cancel{(\neg x_1)} \wedge (\neg x_2) \wedge (x_3) \wedge \\ & (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge \\ & \cancel{(\neg x_1 \vee x_2)} \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge \cancel{(\neg x_1 \vee \neg x_3)} \wedge \\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_2 \;=\; & (\neg x_2) \wedge (x_3) \wedge \\ & (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge \\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (T, T, T)$ hace que g sea $True$ y h_2 sea $False$.

A: (Se eliminan las cláusulas de h_2 que son $False$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{aligned} h_2 \;=\; & \cancel{(\neg x_2)} \wedge (x_3) \wedge \\ & (x_1 \vee \neg x_2) \wedge (x_1 \vee x_3) \wedge \\ & \cancel{(\neg x_1 \vee \neg x_2)} \wedge (\neg x_1 \vee x_3) \wedge \\ & (x_2 \vee x_3) \wedge (\neg x_2 \vee x_3) \wedge \cancel{(\neg x_2 \vee \neg x_3)} \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_3 \quad = \quad & (x_3) \wedge \\ & (x_1 \vee \neg x_2) \ \wedge \ (x_1 \vee x_3) \wedge \\ & (\neg x_1 \vee x_3) \wedge \\ & (x_2 \vee x_3) \ \wedge \ (\neg x_2 \vee x_3) \end{aligned}$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (F, F, F)$ hace que g sea $True$ y h_3 sea $False$.

A: (Se eliminan las cláusulas de h_3 que son $False$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$\begin{aligned} h_3 \quad = \quad & \cancel{(x_3)} \wedge \\ & (x_1 \vee \neg x_2) \ \wedge \ \cancel{(x_1 \vee x_3)} \wedge \\ & (\neg x_1 \vee x_3) \wedge \\ & \cancel{(x_2 \vee x_3)} \ \wedge \ (\neg x_2 \vee x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_4 \quad = \quad & (x_1 \vee \neg x_2) \wedge \\ & (\neg x_1 \vee x_3) \wedge \\ & (\neg x_2 \vee x_3) \end{aligned}$$

¿$h_4 \leftrightarrow g$?

U: No. $v_5 = (F, T, F)$ hace que g sea $True$ y h_4 sea $False$.

A: (Se eliminan las cláusulas de h_4 que son $False$ con v_5 para obtener una fórmula h_5 que tenga el mismo valor que g para la valoración v_5)

$$\begin{aligned} h_4 \quad = \quad & \cancel{(x_1 \vee \neg x_2)} \wedge \\ & (\neg x_1 \vee x_3) \wedge \\ & \cancel{(\neg x_2 \vee x_3)} \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$h_5 \quad = \quad (\neg x_1 \vee x_3)$$

¿$h_5 \leftrightarrow g$?

U: No. $v_6 = (T, F, F)$ hace que g sea $True$ y h_5 sea $False$.

A: (Se eliminan las cláusulas de h_5 que son $False$ con v_6 para obtener una fórmula h_6 que tenga el mismo valor que g para la valoración v_6)

$$h_5 \quad = \quad \cancel{(\neg x_1 \vee x_3)}$$

Por lo tanto, la nueva hipótesis es:

$$h_6 \quad = \quad True \text{ (Conjunción vacía)}$$

¿$h_6 \leftrightarrow g$?

U: Sí.

2.4.5.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $() \wedge (x_1) \wedge (\neg x_1) \wedge \ldots$. El valor de la cláusula vacía $()$ es $False$ y, por equivalencias de la lógica, tenemos que $False \wedge \gamma \equiv False$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \wedge (\neg x_1)$ y se cumple que $\delta \wedge \neg\delta \equiv False$ para cualquier fórmula lógica δ. Por tanto, h_0 es $False$ por más de un motivo.

Al final ha quedado una k-CNF h_6 que es vacía y la conjunción vacía es $True$.

2.4.6 Ejemplo 6 de k-CNF: $k = 0$ y $n = 0$ (True – Conjunción vacía)

2.4.6.1 Enunciado

El usuario tiene en mente una fórmula 0-CNF g que no hace uso de ninguna variable. Por tanto, tenemos que $k = 0$ y $n = 0$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-CNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-CNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = ()$ (Valoración vacía, porque en la fórmula g no hay ninguna variable)

2.4.6.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 0$ y $n = 0$

A: Propuesta inicial:

$$h_0 \quad = \quad ()$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = ()$ hace que g sea $True$ y h_0 sea $False$.

A: (Se eliminan las cláusulas de h_0 que son $False$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$h_0 \quad = \quad \not{\emptyset}$$

Por lo tanto, la nueva hipótesis es:

$$h_1 \quad = \quad True$$

¿$h_1 \leftrightarrow g$?

U: Sí.

2.4.6.3 Comentarios

Nótese que el valor de h_0 es $False$ porque tiene la forma $()$ donde $()$ representa la cláusula vacía, cuyo valor es $False$.

Al final ha quedado una 0-CNF h_1 que es vacía y la conjunción vacía de cláusulas es $True$.

2.5.

Algoritmo de aprendizaje o adivinación para las k-DNF

En este apartado se presenta un algoritmo que es capaz de adivinar o aprender una fórmula proposicional k-DNF a base de hacer preguntas o de pedir pistas al usuario.

El proceso llevado a cabo por el algoritmo puede ser interpretado como adivinación o como aprendizaje de una fórmula:

- Adivinación porque a partir de unas pistas —valoraciones que hacen que la fórmula que el usuario tiene en mente sea falsa— obtiene la fórmula proposicional. Es decir, adivina o averigua cuál es la fórmula.
- Aprendizaje porque a partir de unos ejemplos —valoraciones que hacen que la fórmula que el usuario tiene en mente sea falsa— obtiene la regla general. Es decir, aprende el funcionamiento general de la fórmula.

2.5.1 Escenario

1. El usuario tiene en mente una k-DNF (a la que llamaremos g) en el que pueden aparecer las variables $x_1, x_2, \ldots, x_n$. Pero no es necesario que todas esas variables aparezcan. Además, tiene que cumplirse $n \geq 0$ y $0 \leq k \leq n$.

   Por ejemplo, si k es 2 y n es 4 el usuario podría tener en mente la siguiente fórmula 2-DNF:

   $$g = (\neg x_3) \vee (x_2 \wedge x_3) \vee (x_4)$$

   en la cual la variable proposicional x_1 no aparece.

2. El algoritmo pide al usuario los valores k y n. Estos son los dos únicos datos iniciales que recibe el algoritmo.

3. El algoritmo empieza con el proceso de adivinación:

3.1. El algoritmo genera la k-DNF inicial más general posible utilizando las variables $x_1, x_2, \ldots, x_n$. Llamaremos h_0 a esa k-DNF inicial. Los términos que conforman h_0 tendrán como máximo, tamaño k, es decir, k literales.

Por ejemplo, si $k = 2$ y $n = 2$, la fórmula más general posible es la siguiente 2-DNF:

$$h_0 = ()\vee(x_1)\vee(\neg x_1)\vee(x_2)\vee(\neg x_2)\vee(x_1\wedge x_2)\vee(x_1\wedge\neg x_2)\vee(\neg x_1\wedge x_2)\vee(\neg x_1\wedge\neg x_2)$$

Es decir, una fórmula en la que aparecen x_1 y x_2 (por ser $n = 2$) y que es la disyunción de todos los términos relevantes de cero literales, de un literal y de dos literales, por ser $k = 2$.

Por equivalencias lógicas, no habría que poner, por ejemplo, el término $(x_2 \wedge x_1)$ porque es equivalente a $(x_1 \wedge x_2)$ y, por tanto, no es relevante, no aporta nada. Tampoco habría que poner, por ejemplo, el término $(x_1 \wedge x_1)$ porque es equivalente a (x_1) y no aportaría nada. Y tampoco habría que poner, por ejemplo, el término $(x_1 \wedge \neg x_1)$ porque es equivalente a $False$ y puesto que $\delta \vee False$ es equivalente a δ el término $(x_1 \wedge \neg x_1)$ no aporta nada.

Si n fuera 3 y k fuera 2, habría que utilizar las variables x_1, x_2 y x_3, por ser $n = 3$, y habría que generar todos los términos relevantes de cero literales, de un literal y de dos literales, por ser $k = 2$:

$$\begin{aligned} h_0 = \; & () \vee \\ & (x_1) \vee (\neg x_1) \vee (x_2) \vee (\neg x_2) \vee (x_3) \vee (\neg x_3) \vee \\ & (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge x_3) \vee (x_2 \wedge \neg x_3) \vee \\ & (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

Si n fuera 5 y k fuera 1, habría que utilizar las variables x_1, x_2, x_3, x_4 y x_5, por ser $n = 5$, y habría que generar todos los términos relevantes de cero literales y de un literal, por ser $k = 1$:

$$h_0 = () \vee (x_1) \vee (\neg x_1) \vee (x_2) \vee (\neg x_2) \vee (x_3) \vee (\neg x_3) \vee (x_4) \vee (\neg x_4) \vee (x_5) \vee (\neg x_5)$$

3.2. El algoritmo presenta h_0 al usuario y le pregunta si h_0 es equivalente a g.

3.3. El usuario responde con un "Sí" o con un "No".

3.3.1. Si la respuesta del usuario ha sido "Sí", ello quiere decir que h_0 y g son equivalentes y termina la adivinación.

3.3.2. Si la respuesta del usuario ha sido "No", ello quiere decir que h_0 y g no son equivalentes y, consecuentemente, el algoritmo pide al usuario una valoración

v_1 que haga falsa a g y cierta a h_0. Utilizando v_1, el algoritmo elimina todos los términos de h_0 que son ciertos con v_1 obteniendo así una nueva propuesta h_1, y vuelve al punto 3.2 donde realiza con h_1 lo indicado ahí para h_0. Se repite el proceso hasta dar con una h_j que sea equivalente a g.

2.5.2 Algoritmo para las k-DNF

En la tabla 2.5.1 de la página 44, se muestra el algoritmo correspondiente al problema de adivinar una k-DNF g que el usuario tiene en mente. Puesto que el algoritmo va obteniendo datos desde el exterior a lo largo del proceso de cálculo, decimos que es un sistema inteligente.

2.5.3 Tipos de preguntas o peticiones al usuario: k-DNF

El algoritmo de las k-DNF realiza dos tipos de preguntas al usuario:

- Pedir una valoración v_{j+1} que hace $False$ a g y $True$ a h_j.
- Preguntar al usuario si h_j es equivalente a g.

2.5.4 Especificación no funcional: k-DNF

En la tabla 2.5.2 de la página 45, se muestra la especificación no funcional correspondiente al problema de adivinar una k-DNF g que el usuario tiene en mente.

Los datos iniciales de entrada para el algoritmo serán n y k. El resultado será una k-DNF h. Pero esa k-DNF h no puede ser descrita utilizando solo los datos iniciales n y k. Dicho de otra forma, la k-DNF h no puede ser descrita en función de los datos iniciales n y k. Debido a ello, la especificación es no funcional.

Para adivinar la k-DNF, el algoritmo irá presentando sus hipótesis h_j al usuario. Cada vez que la hipótesis h_j presentada no sea equivalente a g, el usuario dará una pista v_{j+1}, es decir, una valoración que hace cierta a h_j pero falsa a g. Como consecuencia de esto, el resultado final h depende también de g y de las pistas o valoraciones que el usuario dé durante el proceso, pero ni g ni las pistas están disponibles para el algoritmo desde el principio.

{Precondición para datos iniciales de entrada:
$n \in \mathbb{N} \wedge n \geq 0 \wedge k \in \mathbb{N} \wedge k \geq 0 \wedge k \leq n$}

{Precondición para la fórmula g (esta fórmula no es conocida por el algoritmo):
$es\_k\_DNF(g)$}

$j := 0;$
$h_j := k\_DNF\_inicial(n, k);$
$equiv := preguntar\_al\_usuario\_si\_la\_formula\_es\_equivalente(h_j);$

mientras $\neg equiv$ **hacer**

$v_{j+1} := pedir\_al\_usuario\_valoracion\_que\_hace\_cierta\_a(h_j);$
$h_{j+1} := eliminar\_terminos\_ciertos(h_j, v_{j+1});$
$j := j + 1;$
$equiv := preguntar\_al\_usuario\_si\_la\_formula\_es\_equivalente(h_j);$

fin mientras

$h := h_j;$

{Postcondición: $es\_k\_DNF(h) \wedge (h \leftrightarrow g)$}

Tabla 2.5.1. Algoritmo que sirve para aprender o adivinar una fórmula k-DNF g.

Especificación no funcional para el problema de adivinar las k-DNF
Descripción de los datos de entrada iniciales n y k:
$n \in \mathbb{N} \wedge n \geq 0 \wedge k \in \mathbb{N} \wedge k \geq 0 \wedge k \leq n$ donde n indica que se pueden utilizar n variables proposicionales $x_1, x_2, \ldots, x_n$ y k indica que los términos que conforman el k-DNF pueden tener, como máximo k literales.
Descripción del resultado h:
h es una fórmula k-DNF y $h \leftrightarrow g$ (es decir, h es equivalente a g) donde g es la fórmula k-DNF que el usuario tiene en mente.

Tabla 2.5.2. Especificación no funcional del problema consistente en aprender o adivinar una fórmula k-DNF g, que el usuario tiene en mente.

2.6.

Ejercicios resueltos para las k-DNF

En este apartado se presentan algunos ejercicios de examen resueltos para el algoritmo de las k-DNF. Por tanto, lo que se hace en estos ejercicios resueltos es lo que hay que hacer en el examen.

El proceso de adivinación se desarrolla como un diálogo entre el algoritmo (A) y el usuario (U). El usuario tiene en mente una fórmula k-DNF g y el algoritmo en ningún momento dispone de esa k-DNF g. El algoritmo presenta su propuesta h_j al usuario y si h_j no es equivalente a g, el usuario le da una pista: una valoración v_{j+1} que hace falsa a g y cierta a h_j. Para tener una propuesta que se parezca más a g, el algoritmo elimina de h_j aquellos términos que son ciertos con v_{j+1} y así construye una nueva propuesta h_{j+1} que se parece más a g porque h_{j+1} será falsa con v_{j+1}. El proceso se repite hasta que el algoritmo llega a tener una propuesta que es equivalente a g.

Al hacer el ejercicio, nuestro papel es el del algoritmo. Tenemos que ir construyendo propuestas h_j y preguntando al usuario si h_j es equivalente a g y, en caso de que no sea equivalente, tenemos que pedir una valoración v_{j+1} que hace falsa a g y cierta a h_j. Pero al desarrollar este diálogo, el usuario no estará. Debido a ello, en el propio enunciado se nos dan las pistas que el usuario debería darnos cada vez que solicitamos una pista. Hay que agotar y utilizar todas las pistas. Esto significa que cada vez que presentamos una propuesta h_j al usuario, si todavía hay pistas no utilizadas, tenemos que suponer que el usuario nos dirá que no es equivalente a g y que, a continuación, tenemos que utilizar la siguiente pista que todavía no ha sido utilizada. La k-DNF que tengamos cuando se hayan agotado todas las pistas, será la k-DNF definitiva.

Teniendo en cuenta lo que se acaba de indicar, en el ejemplo 1 que viene a continuación habrá que construir una propuesta inicial h_0 y cuatro propuestas h_1, h_2, h_3 y h_4, cada una de las cuales se obtendrá a partir de la propuesta anterior y utilizando la valoración correspondiente. Por ejemplo, h_3 se construirá a aprtir de h_2 y v_3. Al haber cuatro valoraciones —v_1, v_2, v_3 y v_4— hay que llegar hasta h_4.

2.6.1 Ejemplo 1 de k-DNF: $k = 2$ y $n = 3$

2.6.1.1 Enunciado

El usuario tiene en mente una fórmula 2-DNF g que puede hacer uso de 3 variables x_1, x_2 y x_3. Por tanto, tenemos que $k = 2$ y $n = 3$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-DNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-DNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, F, T)$
- $v_2 = (F, T, T)$
- $v_3 = (F, T, F)$
- $v_4 = (F, F, T)$

2.6.1.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 3$

A: Propuesta inicial.

$$\begin{aligned} h_0 \;=\; & () \vee (x_1) \vee (\neg x_1) \vee (x_2) \vee (\neg x_2) \vee (x_3) \vee (\neg x_3) \vee \\ & (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge x_3) \vee (x_2 \wedge \neg x_3) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, F, T)$ hace que g sea $False$ y h_0 sea $True$.

A: (Se eliminan los términos de h_0 que son $True$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \;=\; & \cancel{()} \vee \cancel{(x_1)} \vee (\neg x_1) \vee (x_2) \vee \cancel{(\neg x_2)} \vee \cancel{(x_3)} \vee (\neg x_3) \vee \\ & (x_1 \wedge x_2) \vee \cancel{(x_1 \wedge \neg x_2)} \vee \cancel{(x_1 \wedge x_3)} \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge x_3) \vee (x_2 \wedge \neg x_3) \vee \cancel{(\neg x_2 \wedge x_3)} \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{aligned} h_1 \quad = \quad & (\neg x_1) \ \vee \ (x_2) \ \vee \ (\neg x_3) \vee \\ & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \ \vee \ (\neg x_1 \wedge \neg x_2) \ \vee \ (\neg x_1 \wedge x_3) \ \vee \ (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge x_3) \ \vee \ (x_2 \wedge \neg x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (F, T, T)$ hace que g sea $False$ y h_1 sea $True$.

A: (Se eliminan los términos de h_1 que son $True$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \quad = \quad & \cancel{(\neg x_1)} \ \vee \ \cancel{(x_2)} \ \vee \ (\neg x_3) \vee \\ & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & \cancel{(\neg x_1 \wedge x_2)} \ \vee \ (\neg x_1 \wedge \neg x_2) \ \vee \ \cancel{(\neg x_1 \wedge x_3)} \ \vee \ (\neg x_1 \wedge \neg x_3) \vee \\ & \cancel{(x_2 \wedge x_3)} \ \vee \ (x_2 \wedge \neg x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

Por lo tanto, la nueva propuesta es la siguiente:

$$\begin{aligned} h_2 \quad = \quad & (\neg x_3) \vee \\ & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \ (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge \neg x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (F, T, F)$ hace que g sea $False$ y h_2 sea $True$.

A: (Se eliminan los términos de h_2 que son $True$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{aligned} h_2 \quad = \quad & \cancel{(\neg x_3)} \vee \\ & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \ \cancel{(\neg x_1 \wedge \neg x_3)} \vee \\ & \cancel{(x_2 \wedge \neg x_3)} \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

Por lo tanto, la nueva propuesta es:

$$\begin{aligned} h_3 \quad = \quad & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge \neg x_2) \vee \\ & (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (F, F, T)$ hace que g sea $False$ y h_3 sea $True$.

A: (Se eliminan los términos de h_3 que son $True$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$\begin{array}{rcl} h_3 & = & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & & \cancel{(\neg x_1 \wedge \neg x_2)} \vee \\ & & (\neg x_2 \wedge \neg x_3) \end{array}$$

Por lo tanto, la nueva hipótesis es:

$$\begin{array}{rcl} h_4 & = & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_3) \vee \\ & & (\neg x_2 \wedge \neg x_3) \end{array}$$

¿$h_4 \leftrightarrow g$?

U: Sí.

2.6.1.3 Comentarios

Nótese que el valor de h_0 es $True$ porque tiene la forma $() \vee (x_1) \vee (\neg x_1) \vee \ldots$. El valor del término vacío $()$ es $True$ y, por equivalencias de la lógica, tenemos que $True \vee \gamma \equiv True$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \vee (\neg x_1)$ y se cumple que $\delta \vee \neg\delta \equiv True$ para cualquier fórmula lógica δ. Por tanto, h_0 es $True$ por más de un motivo.

2.6.2 Ejemplo 2 de k-DNF: $k = 2$ y $n = 3$ (simplificable)

2.6.2.1 Enunciado

El usuario tiene en mente una fórmula 2-DNF g que puede hacer uso de 3 variables x_1, x_2 y x_3. Por tanto, tenemos que $k = 2$ y $n = 3$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-DNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-DNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, T, T)$
- $v_2 = (F, T, F)$
- $v_3 = (F, T, T)$
- $v_4 = (T, T, F)$

2.6.2.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 3$.

A: Propuesta inicial.

$$\begin{aligned} h_0 \;=\; & () \vee (x_1) \vee (\neg x_1) \vee (x_2) \vee (\neg x_2) \vee (x_3) \vee (\neg x_3) \vee \\ & (x_1 \wedge x_2) \vee (x_1 \wedge \neg x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge x_3) \vee (x_2 \wedge \neg x_3) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, T, T)$ hace que g sea $False$ y h_0 sea $True$.

A: (Se eliminan los términos de h_0 que son $True$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \;=\; & \cancel{()} \vee \cancel{(x_1)} \vee (\neg x_1) \vee \cancel{(x_2)} \vee (\neg x_2) \vee \cancel{(x_3)} \vee (\neg x_3) \vee \\ & \cancel{(x_1 \wedge x_2)} \vee (x_1 \wedge \neg x_2) \vee \cancel{(x_1 \wedge x_3)} \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & \cancel{(x_2 \wedge x_3)} \vee (x_2 \wedge \neg x_3) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

$$\begin{aligned} h_1 \;=\; & (\neg x_1) \vee (\neg x_2) \vee (\neg x_3) \vee \\ & (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_3) \vee \\ & (\neg x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee (\neg x_1 \wedge \neg x_3) \vee \\ & (x_2 \wedge \neg x_3) \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (F, T, F)$ hace que g sea $False$ y h_1 sea $True$.

A: (Se eliminan los términos de h_1 que son $True$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{aligned} h_1 \;=\; & \cancel{(\neg x_1)} \vee (\neg x_2) \vee \cancel{(\neg x_3)} \vee \\ & (x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_3) \vee \\ & \cancel{(\neg x_1 \wedge x_2)} \vee (\neg x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_3) \vee \cancel{(\neg x_1 \wedge \neg x_3)} \vee \\ & \cancel{(x_2 \wedge \neg x_3)} \vee (\neg x_2 \wedge x_3) \vee (\neg x_2 \wedge \neg x_3) \end{aligned}$$

$$\begin{aligned} h_2 \quad = \quad & (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \ (x_1 \wedge \neg x_3) \ \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \ (\neg x_1 \wedge x_3) \ \vee \\ & (\neg x_2 \wedge x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (F, T, T)$ hace que g sea $False$ y h_2 sea $True$.

A: (Se eliminan los términos de h_2 que son $True$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{aligned} h_2 \quad = \quad & (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \ (x_1 \wedge \neg x_3) \ \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \ \cancel{(\neg x_1 \wedge x_3)} \ \vee \\ & (\neg x_2 \wedge x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

$$\begin{aligned} h_3 \quad = \quad & (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \ (x_1 \wedge \neg x_3) \ \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_2 \wedge x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (T, T, F)$ hace que g sea $False$ y h_3 sea $True$.

A: (Se eliminan los términos de h_3 que son $True$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$\begin{aligned} h_3 \quad = \quad & (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \ \cancel{(x_1 \wedge \neg x_3)} \ \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_2 \wedge x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

$$\begin{aligned} h_4 \quad = \quad & (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_2 \wedge x_3) \ \vee \ (\neg x_2 \wedge \neg x_3) \end{aligned}$$

¿$h_4 \leftrightarrow g$?

U: Sí.

2.6.2.3 Comentarios

Nótese que el valor de h_0 es $True$ porque tiene la forma $() \vee (x_1) \vee (\neg x_1) \vee \ldots$. El valor del término vacío $()$ es $True$ y, por equivalencias de la lógica, tenemos que $True \vee \gamma \equiv True$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \vee (\neg x_1)$ y se cumple que $\delta \vee \neg\delta \equiv True$ para cualquier fórmula lógica δ. Por tanto, h_0 es $True$ por más de un motivo.

La k-DNF h_4 que se ha obtenido al final es equivalente a $(\neg x_2)$. Por tanto, el algoritmo no obtiene la versión más sencilla de la fórmula pero eso no importa. Nos tenemos que quedar con la fórmula que ha obtenido el algoritmo y no tenemos que preocuparnos de simplificarla.

2.6.3 Ejemplo 3 de k-DNF: $k = 1$ y $n = 5$

2.6.3.1 Enunciado

El usuario tiene en mente una fórmula 1-DNF g que puede hacer uso de 5 variables x_1, x_2, x_3, x_4 y x_5. Por tanto, tenemos que $k = 1$ y $n = 5$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-DNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-DNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, T, T, T, T)$
- $v_2 = (T, F, T, F, T)$

2.6.3.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 1$ y $n = 5$.

A: Propuesta inicial.

$$h_0 = \quad () \vee (x_1) \vee (\neg x_1) \vee (x_2) \vee (\neg x_2) \vee (x_3) \vee (\neg x_3) \vee (x_4) \vee (\neg x_4) \vee (x_5) \vee (\neg x_5)$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, T, T, T, T)$ hace que g sea $False$ y h_0 sea $True$.

A: (Se eliminan los términos de h_0 que son $True$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$h_0 = \cancel{()} \vee \cancel{(x_1)} \vee (\neg x_1) \vee \cancel{(x_2)} \vee (\neg x_2) \vee \cancel{(x_3)} \vee (\neg x_3) \vee \cancel{(x_4)} \vee (\neg x_4) \vee \cancel{(x_5)} \vee (\neg x_5)$$

$$h_1 = (\neg x_1) \vee (\neg x_2) \vee (\neg x_3) \vee (\neg x_4) \vee (\neg x_5)$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (T, F, T, F, T)$ hace que g sea $False$ y h_1 sea $True$.

A: (Se eliminan los términos de h_1 que son $True$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$h_1 = (\neg x_1) \vee \cancel{(\neg x_2)} \vee (\neg x_3) \vee \cancel{(\neg x_4)} \vee (\neg x_5)$$

$$h_2 = (\neg x_1) \vee (\neg x_3) \vee (\neg x_5)$$

¿$h_2 \leftrightarrow g$?

U: Sí.

2.6.3.3 Comentarios

Nótese que el valor de h_0 es $True$ porque tiene la forma $() \vee (x_1) \vee (\neg x_1) \vee \ldots$. El valor del término vacío $()$ es $True$ y, por equivalencias de la lógica, tenemos que $True \vee \gamma \equiv True$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \vee (\neg x_1)$ y se cumple que $\delta \vee \neg\delta \equiv True$ para cualquier fórmula lógica δ. Por tanto, h_0 es $True$ por más de un motivo.

2.6.4 Ejemplo 4 de k-DNF: $k = 2$ y $n = 2$ ($False$ – Disyunción vacía)

2.6.4.1 Enunciado

El usuario tiene en mente una fórmula 2-DNF g que puede hacer uso de 2 variables x_1 y x_2. Por tanto, tenemos que $k = 2$ y $n = 2$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-DNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una k-DNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = (T, T)$
- $v_2 = (F, F)$
- $v_3 = (T, F)$
- $v_4 = (F, T)$

2.6.4.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 2$ y $n = 2$.

A: Propuesta inicial.

$$\begin{aligned} h_0 \quad = \quad & (x_1) \ \vee \ (\neg x_1) \ \vee \ (x_2) \ \vee \ (\neg x_2) \ \vee \\ & (x_1 \wedge x_2) \ \vee \ (x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_1 \wedge x_2) \ \vee \ (\neg x_1 \wedge \neg x_2) \end{aligned}$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = (T, T)$ hace que g sea $False$ y h_0 sea $True$.

A: (Se eliminan los términos de h_0 que son $True$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$\begin{aligned} h_0 \quad = \quad & \cancel{(x_1)} \ \vee \ (\neg x_1) \ \vee \ \cancel{(x_2)} \ \vee \ (\neg x_2) \ \vee \\ & \cancel{(x_1 \wedge x_2)} \ \vee \ (x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_1 \wedge x_2) \ \vee \ (\neg x_1 \wedge \neg x_2) \end{aligned}$$

La nueva propuesta es la siguiente:

$$\begin{aligned} h_1 \quad = \quad & (\neg x_1) \ \vee \ (\neg x_2) \ \vee \\ & (x_1 \wedge \neg x_2) \ \vee \\ & (\neg x_1 \wedge x_2) \ \vee \ (\neg x_1 \wedge \neg x_2) \end{aligned}$$

¿$h_1 \leftrightarrow g$?

U: No. $v_2 = (F, F)$ hace que g sea $False$ y h_1 sea $True$.

A: (Se eliminan los términos de h_1 que son $True$ con v_2 para obtener una fórmula h_2 que tenga el mismo valor que g para la valoración v_2)

$$\begin{array}{rcl} h_1 & = & \cancel{(\neg x_1)} \vee \cancel{(\neg x_2)} \vee \\ & & (x_1 \wedge \neg x_2) \vee \\ & & (\neg x_1 \wedge x_2) \vee \cancel{(\neg x_1 \wedge \neg x_2)} \end{array}$$

La nueva propuesta es la siguiente:

$$\begin{array}{rcl} h_2 & = & (x_1 \wedge \neg x_2) \vee \\ & & (\neg x_1 \wedge x_2) \end{array}$$

¿$h_2 \leftrightarrow g$?

U: No. $v_3 = (T, F)$ hace que g sea $False$ y h_2 sea $True$.

A: (Se eliminan los términos de h_2 que son $True$ con v_3 para obtener una fórmula h_3 que tenga el mismo valor que g para la valoración v_3)

$$\begin{array}{rcl} h_2 & = & \cancel{(x_1 \wedge \neg x_2)} \vee \\ & & (\neg x_1 \wedge x_2) \end{array}$$

La nueva propuesta es la siguiente:

$$h_3 = (\neg x_1 \wedge x_2)$$

¿$h_3 \leftrightarrow g$?

U: No. $v_4 = (F, T)$ hace que g sea $False$ y h_3 sea $True$.

A: (Se eliminan los términos de h_3 que son $True$ con v_4 para obtener una fórmula h_4 que tenga el mismo valor que g para la valoración v_4)

$$h_3 = \cancel{(\neg x_1 \wedge x_2)}$$

La nueva propuesta es la siguiente:

$$h_4 = False \text{ (disyunción vacía)}$$

¿$h_4 \leftrightarrow g$?

U: Sí.

2.6.4.3 Comentarios

Nótese que el valor de h_0 es $True$ porque tiene la forma $() \vee (x_1) \vee (\neg x_1) \vee \ldots$. El valor del término vacío $()$ es $True$ y, por equivalencias de la lógica, tenemos que $True \vee \gamma \equiv True$ para cualquier fórmula lógica γ. Además, h_0 contiene la subfórmula $(x_1) \vee (\neg x_1)$ y se cumple que $\delta \vee \neg\delta \equiv True$ para cualquier fórmula lógica δ. Por tanto, h_0 es $True$ por más de un motivo.

La k-DNF resultante h_4 es vacía y la disyunción vacía es $False$.

2.6.5 Ejemplo 5 de k-DNF: $k = 0$ y $n = 0$ ($False$ – Disyunción vacía)

2.6.5.1 Enunciado

El usuario tiene en mente una fórmula 0-DNF g que no utiliza ninguna variable. Por tanto, tenemos que $k = 0$ y $n = 0$. Hay que aplicar paso a paso el algoritmo de aprendizaje de fórmulas k-DNF, indicando las preguntas que el algoritmo realiza al usuario y las distintas propuestas h_j que el algoritmo va construyendo hasta obtener una 0-DNF equivalente a g.

Los contraejemplos v_{j+1} que el usuario va dando sucesivamente cada vez que la propuesta h_j no es equivalente a la fórmula objetivo g son los siguientes:

- $v_1 = ()$ (Valoración vacía, porque en la fórmula g no hay ninguna variable)

2.6.5.2 Solución

A continuación se muestra el diálogo entre el algoritmo (A) y el usuario (U). Se escribirá T y F en vez de $True$ y $False$:

A: ¿Cuál es el valor de k y de n?

U: $k = 0$ y $n = 0$.

A: Propuesta inicial.

$$h_0 \quad = \quad ()$$

¿$h_0 \leftrightarrow g$?

U: No. $v_1 = ()$ hace que g sea $False$ y h_0 sea $True$.

A: (Se eliminan los términos de h_0 que son $True$ con v_1 para obtener una fórmula h_1 que tenga el mismo valor que g para la valoración v_1)

$$h_0 \quad = \quad \not{()}$$

La nueva propuesta es la siguiente:

$$h_1 \quad = \quad False$$

¿$h_1 \leftrightarrow g$?

U: Sí.

2.6.5.3 Comentarios

Nótese que el valor de h_0 es $True$ porque tiene la forma $()$, donde $()$ representa el término vacío, cuyo valor es $True$.

Al final ha quedado una 0-DNF h_1 que es vacía y la disyunción vacía es $False$.

2.7.

Computabilidad y sistemas inteligentes

En los apartados anteriores se han presentado dos algoritmos que adivinan —con la ayuda de pistas— la fórmula que el usuario tiene en mente. Esos algoritmos no admiten una especificación funcional porque el resultado final no es descriptible utilizando solo los datos de entrada iniciales n y k. Los algoritmos que tienen esa característica reciben el nombre de "Sistemas Inteligentes". Son algoritmos "vivos" o "espabilados" o "hábiles" que se las arreglan para obtener información de la que a priori no disponen. Para ello, han de contactar o interactuar con uno o varios oráculos, que serán las fuentes de las cuales obtendrán esa información adicional que necesitan.

Volviendo al caso de los algoritmos que adivinan las k-CNF y las k-DNF que el usuario tiene en mente, lo primero que hay que decir es que es imposible construir un programa que sirva para adivinar una k-CNF o una k-DNF si la única información disponible durante todo el proceso de cálculo es el valor de k y el valor de n. Por tanto, podríamos decir que esos dos problemas de adivinación son incomputables. Pero la situación cambia si permitimos que los algoritmos contacten con un oráculo (el usuario u otra fuente de información) que les va aportanto la información adicional imprescindible para realizar la adivinación.

Ahí podemos constatar que al permitir que el algoritmo acceda a oráculos externos, un problema pasa de incomputable a computable. Como no sabemos qué tipos de oráculos pueden existir en el universo, no tiene sentido —al menos en las primeras etapas del estudio de la computabilidad— analizar la computabilidad de los algoritmos que tienen acceso a oráculos, porque puede haber oráculos muy muy avanzados que para nosotros desconocemos. Esos oráculos avanzados podrían convertir en computable un problema que para nosotros e incomputable.

Debido a lo que se acaba de exponer, nuestro estudio de la computabilidad se va a ceñir solo a problemas que admiten una especificación funcional. Es decir, algoritmos que reciben unos datos de entrada iniciales y que a partir de esos datos iniciales, obtienen el resultado final, sin ningún contacto con ningún oráculo y sin recibir ninguna información adicional durante el proceso de cálculo.