Diseño y verificación formal de programas moleculares en el modelo débil de Amos

Sergio Rodríguez Calvo

Septiembre de 2017

Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Abstract. *En el presente trabajo se pretende estudiar el diseño y la verificación formal de dos programas moleculares en el modelo débil de Amos, que resuelven el problema de la generación de permutaciones; y el problema del camino hamiltoniano en su versión dirigida y sin nodos distinguidos. En concreto, el presente trabajo se centra en reescribir, así como, completar algunos detalles de las demostraciones, que se encuentran el trabajo original [1]. El desarrollo de este trabajo tiene como objetivo ser entregado como trabajo final de la asignatura de Computación Bioinspirada.*

1. Introducción

En la década de los cincuenta comienza a ser evidente que existe una analogía entre algunos procesos matemáticos y ciertos procesos biológicos. Un organismo vivo puede ser visto como el resultado de aplicar una serie de operaciones bioquímicas sobre una cadena de ácido desoxirribonucleico (ADN).

Posteriormente, en la decada de los noventa, se demostró que se pueden usar ciertos procesos biológicos para atacar la resolubilidad de problemas matemáticos difíciles. Estos problemas también son conocidos como computacionalmente intratables, y son aquellos que su solución algorítmica toma una cantidad de recursos exponenciales en el tamaño del dato de entrada.

Esta resolubilidad está relacionada con la potencia de cálculo y la densisdad de almacenamiento de los ordenadores convencionales.

En la propia década de los cincuenta ya se introdujo el concepto teórico de computación a nivel molecular. En los ordenadores convencionales la paralelización y la miniaturización son un objetivo importante, y la computación molecular puede suponer un paso más en este sentido.

Sobre todo, a partir de que en la década de los ochenta, cuando se demostró la existencia de un límite en la potencia de cálculo y, también, en la miniaturización de los componentes electrónicos empleados en los ordenadores convencionales.

Por último, se enumeran las principales ventajas del uso de la computación molecular:

- Sustitución de la luz por reacciones químicas, lo que implica un ahorro del consumo energético.
- El uso de interruptores moleculares permite, según se estima, disponer de más de mil procesadores en el mismo espacio que un procesador convencional.
- Se estima que los interruptores moleculares pueden aumentar cien mil millones de veces la capacidad de procesamiento respecto a los ordenadores convencionales.
- Se estima que se podría reproducir la capacidad de cien ordenadores en el tamaño de un grano de sal fina.

2. Modelo débil de Amos

Antes de introducir el Modelo débil de Amos, se necesita previamente conocer los detalles y principios de la computación molecular, así como, los distintos modelos previos a este, los cuales se pueden encontrar en el documento original del profesor del departamento de Ciencias de la Computación de la Universidad de Sevilla, Mario de Jesús Pérez Jiménez [1].

El Modelo débil de Amos consiste en un modelo de computación basada en ADN, esto es, que utiliza como sustrato computacional el ADN, y en el cual se realizan filtrados sobre el sustrato anterior. En este caso, no existe memoria de acceso aleatorio como en la computación clásica. Para almacenar el sustrato, al igual que en otros modelos de computación molecular, se utiliza un tubo de ensayo que contendrá la muestra.

Dicho tubo es un multiconjunto finito de cadenas del alfabeto $\sum_{ADN} = \{A, C, G, T\}$.

A nivel abstracto, en el Modelo débil de Amos las operaciones que se pueden realizar sobre los tubos son las siguientes:

- **Quitar (T,{$\gamma_1, ..., \gamma_n$}):** Dado un tubo, T, y un número finito de cadenas, $\gamma_1, ..., \gamma_n$, de $\sum$, devuelve el tubo obtenido de T eliminando todas aquellas cadenas que contengan, al menos, una ocurrencia de alguna de las cadenas $\gamma_1, ..., \gamma_n$.
- **Copiar (T,{$T_1, ..., T_n$}):** Dado un tubo, T, y un numero natural $k \geq 2$, devuelve k tubos, $T_1, ..., T_n$, que son copias exactas de T.
- **Unión ({$T_1, ..., T_n$}):** Dados los tubos $T_1, ..., T_n$, con $k \geq 2$, devuelve un tubo T, cuyo contenido es la unión de los tubos $T_1, ..., T_n$ como multiconjuntos.
- **Selección (T):** Dado un tubo, T, selecciona aleatoriamente un elemento de T en el caso en que $T \neq \emptyset$; en caso contratio, devuelve **NO**.

Estas operaciones serán instrucciones moleculares primitivas del modelo

débil. Además, cabe destacar que en este modelo la única operación molecular que implementa paralelismo masivo es *quitar*.

El primer problema, el de la generación de permutaciones, será abordado previo al problema del camino hamiltoniano , ya que, será necesario para el modelado y resolución del segundo problema.

2.1. Problema de la generación de permutaciones

En esta sección se muestra como abordar desde el punto de vista de la computación molecular la resolubilidad del problema de la generación de permutaciones. Antes, se define en qué consiste una permutación para, a continuación, abordar el problema.

Una permutación la definimos como *dado un numero natural, $n \geq 1$, una permutación de orden n es una aplicación biyectiva del conjunto finito {1,...,n} en sí mismo.*

Una vez tenemos definido qué es una permutación vamos a introducir el problema de generación de permutaciones, que consiste en, *dado un número natural, $n \geq 2$, generar todas las permutaciones de orden n.*

2.1.1. Diseño del programa molecular

En primer lugar, hay que definir el modelo mediante el cual se pretende representar el problema con los elementos que tenemos en la computación molecular. Esto es, definir por un lado el alfabeto: $\sum = \{(p_i, c_j) : 1 \leq i, j \leq n\}$, donde p_i y c_j son dos oligos que codificarán respectivamente la posición i-ésima en la permutación y el número j.

A continuación, se necesita definir el tubo de entrada, T_0, para poder comenzar el experimento, es decir, aplicar las operaciones abstractas de la sección anterior según un determinado algoritmo. En este caso, se trata de un multiconjunto de entrada con un número finito de moléculas y que codifican

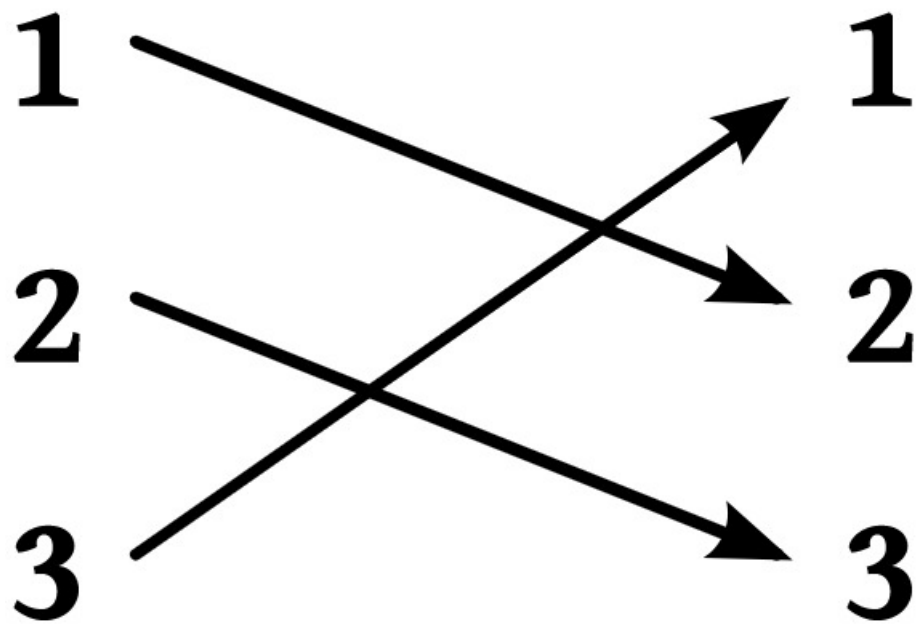

Figura 1: Ejemplo de permutación considerada como función biyectiva. En este caso, para $n = 3$.

todas las posibles sucesiones para una longitud n. Formalmente sería:

$$T_0 = \{\{\sigma \in \sum^{n} : \exists x_1, ..., \exists x_n(\sigma = (p_1, x_1), (p_2, x_2), ..., (p_n, x_n))\}\}$$

En la fórmula anterior, σ, es una codificación concreta, tal que, $\sigma = p_1x_1...p_nx_n$.

La idea aquí es generar todas las posibles permutaciones, utilizando un tubo inicial, T_0, que contiene la codificación de todas las posibles soluciones de longitud n. El programa sigue los siguientes pasos hasta $n - 1$ veces:

- Un primer filtro sobre T_0 para seleccionar las moleculas, σ, tal que:

$$\forall r > 1((\sigma)_1 \neq (\sigma)_r)$$

  Esto es, para todas las posiciones, r, mayores que 1, devolver todas aquellas codificaciones, tal que, el número que ocupa la posición r-ésima sea distinto del número de la primera posición.

- Un segundo filtro respecto del paso anterior donde se seleccionan las moleculas, σ, tal que:

$$\forall r > 2((\sigma)_1 \neq (\sigma)_r \wedge (\sigma)_2 \neq (\sigma)_r)$$

  Esto es, del conjunto resultante del paso anterior, para todas las posiciones, r, mayores que dos, devolver todas aquellas codificaciones, tal que, el número que ocupa la posición r-ésima sea distinto de el número de la posición primera y de la posición segunda a la vez.

La notación, $(\sigma)_r$, representa el número que ocupa la posición *r-ésima* en la sucesión de longitud n codificada por σ. Es decir, $(\sigma)_r = x_r$ para todo r $(1 \leq r \leq n)$.

El algoritmo a seguir en el programa molecular, basado en la idea de filtrar sobre un determinado tubo T_0 de entrada, es el siguiente:

Require: T_0
  for $j \leftarrow 1$ in $n-1$ **do**
    copiar$(T_0, \{T_1, ..., T_n\})$
    for $i \leftarrow 1$ in n **do**
      quitar$(T_i, \{p_j r : r \neq i\} \cup \{p_k i : j+1 \leq k \leq n\})$
    unión$(\{T_1, ..., T_n\}, T_0)$
  return T_0

Este algoritmo presenta una complejidad $O(n^2)$, es decir, orden cuadrático en n. Por tanto, el número de operaciones moleculares es n^2.

2.1.2. Metodología para verificar formalmente programas con bucle principal

En esta sección, se explica una metogología para verificar formalmente que un programa con bucle principal satisface una propiedad.

Se representa el bucle principal como una fórmila y se busca que dicha fórmula satisfaga las siguientes propiedades:

- La fórmila es verdadera antes de comenzar el bucle.
- La fórmula es invariante, es decir, que es verdadera también al finalizar el bucle.
- La veracidad indica que satisface la propiedad requerida.

Además, se necesita probar la corrección y completitud.

Con la demostración de la correción se pretende demostrar las propiedades de una fórmula, la cual modela el programa. Tal y como se describió previamente. Consiste en un teorema (invarianza) y un corolario (correción), donde se prueba que toda molécula del tubo de salida codifica una valoración que hace verdadera dicha fórmula.

Con la demostración de la completitud se pretende demostrar las propiedaes de la fórmula, la cual modela el programa. Tal y como se describió previamente. Consiste, también, en un teorema (invarianza) y un corolario (completitud), donde se prueba que toda molécula del tubo de salida, y que codifica una valoración que hace verdadera la fórmula, aparecerá en el tubo de salida.

2.1.3. Verificación formal del programa molecular

Para realizar la verificación formal es necesario etiquetar cada uno de los tubos que se van a obtener a los largo del algoritmo descrito anteriormente. Por tanto, el algoritmo va a ser reescrito de la siguiente forma:

Require: T_0
  for $j \leftarrow 1$ in $n-1$ **do**
    copiar($T_0^{j-1}, \{T_1^j, ..., T_n^j\}$)
    for $i \leftarrow 1$ in n **do**
      $\bar{T}_i^j \leftarrow quitar(\mathrm{T}_i^j, \{p_j r : r \neq i\} \cup \{p_k i : j+1 \leq k \leq n\})$
    unión($\{\bar{T}_1, ..., \bar{T}_n\}, T^j$)
  return T_{n-1}

Antes de continuar, hay que aclarar la notación a emplear a partir de aquí. $A_{\sigma,j}$ será el conjunto $\{\sigma_1, ..., \sigma_j\}$, es decir, número en cada posición j-ésima, para $(1 \leq j \leq n)$, en cada $\sigma \in T_0$. Esto es, un conjunto que contiene todos los números que contienen una determinada codificación, σ, desde 1 hasta la posición j en cada caso.

Esta notación que se acaba de describir será necesaria para la correción formal del programa que se acaba de reescribir, y que se utiliza en la siguiente formula:

$$\theta(j) \equiv \forall \sigma \in T^j(|A_{\sigma,j}| = j \wedge \forall r(j+1 \leq r \leq n \longrightarrow (\sigma)_r \notin A_{\sigma,j}))$$

La formula, $\theta(j)$, expresa que toda molecula del tubo T^j codifica una sucesión de longitud n tal que los j primeros términos son distintos entre sí y, además, distintos de los restantes términos de la sucesión.

A continuación, se van a exponer una serie de teoremas con los que se pretende realizar la verificación formal, tomando el algoritmo que se ha reescrito en este apartado.

Teorema 1.1. $\forall j(1 \leq j \leq n-1 \longrightarrow \theta(j))$. Es decir, la formula θ es un invariante del bucle principal. Esto es, que θ no cambia a lo largo de las transformaciones que sufre el tubo en el bucle principal.

Dicho de otro modo, para todo valor j, comprendido entre 1 y $n-1$ ambos inclusive, se cumple $\theta(j)$, por tanto, $\theta(j)$ no varía a lo largo de la ejecución del algoritmo.

Se va a realizar la demostración por inducción débil sobre j, y esta es:

En primer lugar, se tiene que demostrar para $j = 1$. Sea $\sigma \in T^1$. Existe $x \in \{1, ..., n\}$ tal que $\sigma \in \bar{T}^1_x = quitar(T^1_x, \{p_1 r : r \neq x\} \cup \{p_k x : 2 \leq k \leq n\})$.

Entonces $A_{\sigma,1} = \{(\sigma)_1\} = \{x\}$, es decir, $|A_{\sigma,1}| = 1$. Ademas, si $2 \leq k \leq n$, entonces $\sigma \in \bar{T}^1_x \Longrightarrow (\sigma)_1 = x \wedge (\sigma)_k \neq x \Longrightarrow (\sigma)_k \notin A_{\sigma,1}$

Esto último quiere decir, que σ existe en el tubo resultado de realizar la eliminación de los elementos que cumplen la condición descrita previamente, que llamaremos tubo complementario. Por tanto, se tiene que el conjunto de valores $A_{\sigma,1}$ contiene el elemento x y ese elemento no está en la posiciones siguientes en el resto de codificaciones de σ.

A continuación, suponemos cierto para j, y tenemos que $j < n-1 (j \geq 1)$. Se va a demostrar para $j + 1$.

Sea $\sigma \in T^{j+1}$. Y sea $x \in \{1, ..., n\}$ tal que

$$\sigma \in \bar{T}^{j+1}_x = quitar(T^{j+1}_x, \{p_{j+1} r : r \neq x\} \cup \{p_k x : j + 2 \leq k \leq n\})$$

Entonces, se verifica que

$$\sigma \in T^{j+1}_x = T_j \wedge (\sigma)_{j+1} = x \wedge \forall k (j + 2 \leq k \leq n \longrightarrow (\sigma)_k \neq x)$$

De la hipotesis de inducción se tiene que

$$|A_{\sigma,j}| = j \wedge \forall r (j + 1 \leq r \leq n \longrightarrow (\sigma)_k \neq x)$$

Teniendo presente que

$$A_{\sigma,j+1} = A_{\sigma,j} \cup \{(\sigma)_{j+1}\}$$

Finalmente si $j+2 \leq k \leq n$, entonces $(\sigma)_k \notin A_{\sigma,j}$ y $(\sigma)_k \neq x = (\sigma)_{j+1}$. Por tanto, se tiene que

$$(\sigma)_k \notin A_{\sigma,j+1}$$

Corolario 1.2. *(Correción del programa) Toda molécula del tubo de salida codifica una permutación de orden n.*

La demostración es, sea $\sigma \in T^{n-1}$. Como la fórmula $\theta(n-1)$ es verdadera resulta que $|A_{\sigma,n-1}| = n-1 \wedge \forall r(n-1+1 \leq r \leq n \longrightarrow (\sigma)_r \notin A_{\sigma,n-1})$. Como $A_{\sigma,n} = A_{\sigma,n-1} \cup \{(\sigma)_n\}$, concluimos que $|A_{\sigma,n}| = n$. Por tanto, la molécula σ codifica una permutación de orden n.

Para establecer la completitud del programa, se va a considerar la siguiente fórmula:

$$\delta(j) \equiv \forall \sigma \in T^0(|A_{\sigma,n}| = n \longrightarrow \sigma \in T^j)$$

Esto es, la fórmula $\delta(j)$ expresa que toda molécula del tubo T^i codifica una sucesión de longitud n tal que los j primeros términos son distintos entre sí y, además, distintos de los restantes términos de la sucesión.

Teorema 1.3. $\forall j(1 \leq j \leq n-1 \longrightarrow \delta(j))$. Es decir, la formula δ es un invariante del bucle principal. Esto es, que δ no cambia a lo largo de las transformaciones que sufre el tubo en el bucle principal.

La demostración, por inducción débil sobre j, es, sea $\sigma \in T^0$ tal que $|A_{\sigma,n}| = n$. Entonces la mólecula σ codifica una permutación de orden n. Se tiene que $\sigma \in T^1_{(\sigma)_1}$ y $\forall r(2 \leq r \leq n \longrightarrow (\sigma)_r \neq (\sigma)_1)$.

Luego, $\sigma \in quitar(T^1_{(\sigma)_1}, \{p_1 r : r \neq (\sigma)_1\} \cup \{p_k(\sigma)_1 : 2 \leq k \leq n\})$. Por tanto

$$\sigma \in \bar{T}^1_{(\sigma)_1} \subseteq T^1$$

Sea $j < n-1 (j \geq 1)$ y supongamos que el resultado es cierto para j.

Sea $\sigma \in T^0$ tal que $|A_{\sigma,n}| = n$. De la hipótesis de inducción resulta que $\sigma \in T^j$. Luego

$$\sigma \in T^{j+1}$$

y

$$\forall r(j+2 \leq r \leq n \longrightarrow (\sigma)_r \neq (\sigma)_{j+1})$$

De donde se deduce que

$$\sigma \in \bar{T}^{j+1}_{(\sigma)_{j+1}} \subseteq T^{j+1}$$

Corolario 1.4. *(Completitud del programa) Toda molécula del tubo de ensayo inicial que codifica una permutación de orden n, está contenida en el tubo de salida.*

Por último, basta tener presente que al finalizar la ejecución del programa molecular, la fórmula $\delta(n-1)$ es verdadera, y que el tubo de salida del programa es T^{n-1}.

2.2. Problema del camino hamiltoniano en versión dirigida sin nodos distinguidos

En este sección se muestra como abordar desde el punto de vista de la computación molecular la resolubilidad del problema del camino hamiltoniano en su versión digidia y sin nodos distinguidos.

El problema del camino hamiltoniano en su versión dirigida y sin nodos

distinguidos consiste en dado un grafo dirigido, determinar si existe un camino simple que pasa por todos los nodos del grafo. O lo que es lo mismo, si el grafo posee un ciclo hamiltoniano.

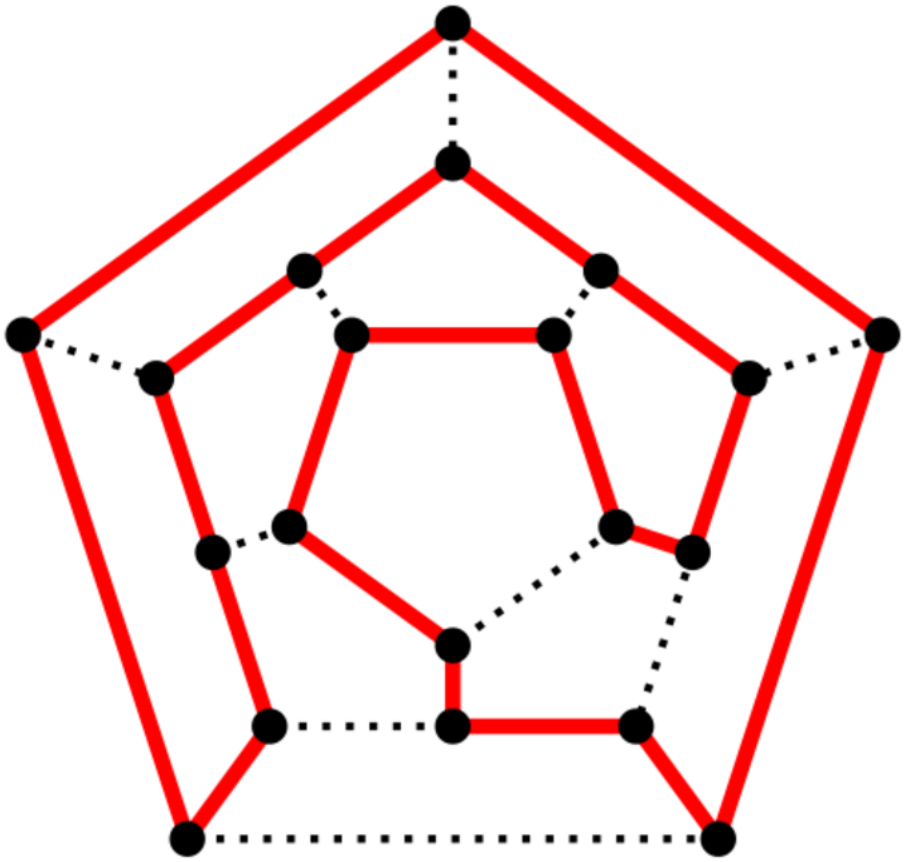

Figura 2: Ejemplo de camino hamiltoniano sobre un grafo.

2.2.1. Diseño del programa molecular

Para dar solución a este problema en el modelo débil de Amos, partimos de un $G = (V, E)$, tal que, G es un grafo dirigido con $V = \{1, ..., n\}$ nodos.

Se debe definir en primer lugar el alfabeto $\sum = \{(p_i, c_j) : 1 \leq i, j \leq n\}$. Donde, p_i y c_j, son dos oligos que codifican respectivamente la posición i-

ésima del camino, y el nodo j. Es decir, el nodo j está en la posición i-ésima del camino.

Como punto de partida, se necesita un tubo de ensayo inicial, T_0. Este problema necesita de todas las permutaciones de orden n, por tanto, el tubo de ensayo inicial es igual que el utilizado en el problema de la generación de permutaciones y una notación parecida. Esto es:

$$T_0 = \{\{\sigma \in \sum^{n} : \exists x_1, ..., \exists x_n(\sigma = (p_1, x_1), (p_2, x_2), ..., (p_n, x_n))\}\}$$

En este caso, la notación es la siguiente: si se tiene $\sigma = p_1x_1p_2x_2...p_nx_n \in T_0$, entonces para cada r $(1 \leq r \leq n)$ se notará $(\sigma)_r = x_r$ y $\sigma_r = ((\sigma)_1, (\sigma)_2, ..., (\sigma)_r)$. Es decir, si σ codifica un camino del grafo, entonces x_r, de $\sigma_r = (x_1, x_2, ..., x_r)$, será el nodo r-ésimo del camino.

Hay que tener en cuenta que si σ codifica una determinada permutación de orden n y, además, codifica un camino del grafo, éste será hamiltoniano.

Los pasos a seguir para resolver este problema, partiendo de un tubo inicial T_0 con todas las posibles permutaciones de orden n, e iterando $n-1$ veces, son los siguientes:

- Se seleccionan las moléculas, σ, tales que σ_2 es un camino de G.
- De éstas, se seleccionan las moleculas, σ, tales que σ_3 es un camino de G, verificando la condición $((\sigma)_2, (\sigma)_3 \in E)$.

El algoritmo será el siguiente:

Require: T_0
  for $i \leftarrow 1$ in $n-1$ **do**
    $T_0 \leftarrow quitar(T_0, \{jp_{i+1}k : (j,k) \notin E\})$
  $seleccionar(T_0)$

La complejidad de este algoritmo es $O(n)$, es decir, tiene un orden de complejidad lineal.

2.2.2. Verificación formal del programa molecular

Para realizar la verificación formal es necesario etiquetar cada uno de los tubos que se van a obtener a los largo del algoritmo descrito anteriormente. Por tanto, el algoritmo va a ser reescrito de la siguiente forma:

Require: T_0
  for $i \leftarrow 1$ in $n-1$ **do**
    $T_i \leftarrow quitar(T_{i-1}, \{jp_{i+1}k : (j,k) \notin E\})$
  $seleccionar(T_{n-1})$

Para poder realizar la verificación formal se va considerar la siguiente fórmula:

$$\theta(i) \equiv \forall \sigma \in T_{i-1} \forall r (1 \leq r \leq i \longrightarrow ((\sigma)_r, (\sigma)_{r+1}) \in E)$$

Esta fórmula, $\theta(i)$, viene a decir que para toda molecula σ del tubo T_{i-1} se verifica que σ_i es un camino del grafo G.

A continuación, se van a exponer una serie de teoremas con los que se pretende realizar la verificación formal, tomando el algoritmo que se ha reescrito en este apartado.

Teorema 2.1. $\forall i (2 \leq i \leq n \longrightarrow \theta(i))$. Es decir, la fórmula θ es un invariante del bucle principal.

La demostración, por inducción débil sobre i es: sea $\sigma \in T_1$. Puesto que $T_1 = quitar(T_0, jp_2k : (j,k) \notin E)$, se deduce que $((\sigma)_1, (\sigma)_2) \in E$.

Sea i tal que $2 \leq i < n$ y supongamos cierto el resultado para i.

Teniendo presente que $T_i = quitar(T_{i-1}, jp_{i+1}k : (j,k) \notin E)$, resulta que

$\sigma \in T_{i-1}$ y $((\sigma)_i, (\sigma)_{i+1}) \in E$. De la hipotesis de inducción se deduce que $\forall r(1 \leq r < i \longrightarrow ((\sigma)_r, (\sigma)_{r+1}) \in E)$. Por tanto, σ_{i+1} es un camino del grafo G.

Corolario 2.2. *(Corrección del programa) Toda molécula del tubo de salida codifica un camino hamiltoniano del grafo G.*

La demostración es, sea $\sigma \in T_{n-1}$. Como la fórmula $\theta(n)$ es verdadera resulta que $\forall r(1 \leq r < n \longrightarrow ((\sigma)_r, (\sigma)_{r+1}) \in E)$. Por tanto, σ_n es un camino de G. Teniendo presente que la molécula σ codifica una permutación de orden n, concluimos que σ codifica un camino hamiltoniano del grafo G.

Para establecer la completitud del programa molecular diseñado, consideremos la siguiente fórmula:

$$\delta(i) \equiv \forall \sigma \in T_0([\forall r(1 \leq r < n \longrightarrow ((\sigma)_r, (\sigma)_{r+1}) \in E)] \longrightarrow \sigma \in T_{i-1})$$

Es decir, la formula $\delta(i)$ expresa que toda molecula del tubo inicial que codifica un camino hamiltoniano de G pertenece al tubo T_{i-1}.

Teorema 2.3. $\forall i(2 \leq i \leq n \longrightarrow \delta(i))$. Es decir, la fórmula δ es un invariante del bucle principal.

La demostración, por inducción débil sobre i es: sea $\sigma \in T_0$ tal que σ codifica un camino hamiltoniano de G. Entonces $\forall r(1 \leq r \leq n \longrightarrow ((\sigma)_r, (\sigma)_{r+1}) \in E)$. Luego, $\sigma \in quitar(T_0, \{jp_2k : (j,k) \notin E\}) = T_1$.

Sea i tal que $2 \leq i < n$ y supongamos cierto el resultado para i.

Sea $\sigma \in T_0$ tal que σ codifica un camino hamiltoniano de G. Por hipotesis de inducción se tiene que $\sigma \in T_{i-1}$. Luego, $\sigma \in quitar(T_{i-1}, \{jp_{i+1}k : (j,k) \notin E\}) = T_i$.

Corolario 2.4. *(Completitud del programa) Toda molécula del tubo de ensayo inicial codifica un camino hamiltoniano de G, está en el tubo de salida.*

Para la demostración basta tener presente que la fórmula $\delta(n)$ es verdadera y que el tubo de salida del programa es T_{n-1}.

Referencias

[1] Mario de Jesús Pérez Jiménez. *Computación molecular sin memoria basada en ADN*. 2001.