NP–Completitud y Computación ADN

Pérez-Jiménez, M.J.; Sancho, F.; Graciani, M.C.; Romero, A.
Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
e-mail: marper@cica.es

Resumen

En este trabajo se presentan dos modelos de computación molecular: el *modelo restringido* de L. Adleman y el *modelo débil* de M. Amos. Se diseñan programas moleculares, basados en ADN, que resuelven el problema **SAT** en el modelo restringido y el problema **3-COL** en el modelo débil, y se establece la verificación formal de los programas diseñados, utilizando la técnica de los invariantes. Finalmente, se describe un procedimiento que permite controlar, en cierto sentido, los errores a que puede dar lugar la ejecución de algunas operaciones de los modelos.

1 Introducción

El **ADN** (ácido desoxirribonucleico) es una molécula fundamental de los seres vivos. En 1953, J. Watson y F. Crick demuestran que las moléculas de ADN codifican toda la información genética de las células vivas y justifican la posibilidad de utilizar ciertas técnicas para la manipulación de las mismas. Las moléculas de ADN se pueden usar como una especie de memoria representando la información a través de una codificación basada en cuatro nucleótidos. Dicha información se puede manipular realizando con las moléculas una serie de reacciones químicas. Así surge la posibilidad de implementar algoritmos a través de la manipulación de moléculas de ADN.

En noviembre de 1994, L. Adleman ([1]) resolvió una instancia concreta del *problema del circuito hamiltoniano* mediante la manipulación de moléculas de ADN usando técnicas de biología molecular. El experimento de Adleman proporciona un primer ejemplo de computación a nivel molecular; muestra nuevas perspectivas de las moléculas de ADN como estructura de datos, ilustra la posibilidad de usar moléculas de ADN para resolver instancias de problemas computacionalmente intratables, y manifiesta la capacidad del ADN para simular computaciones de forma masivamente paralela.

En el experimento de Adleman se parte de un tubo de ensayo inicial que contiene moléculas de ADN codificando todas las posibles soluciones del problema y se eliminan aquellas que no son correctas. Con este experimento nace propiamente la computación ADN y se ponen de manifiesto las presumibles ventajas de esta computación respecto de los ordenadores convencionales, en lo que respecta a velocidad de cálculo, consumo de energía y densidad de información. No obstante, el citado experimento no justifica la posibilidad de resolver, ni siquiera teóricamente, cualquier instancia del problema del circuito hamiltoniano.

En abril de 1995, R.J. Lipton ([5]) resuelve una instancia del problema **SAT** de la *satisfactibilidad de la Lógica Proposicional* siguiendo las ideas de Adleman, si bien en el experimento de Lipton se considera un tubo de ensayo inicial que **no** depende del dato de entrada concreto, sino únicamente de su "tamaño".

Así se van sucediendo experimentos basados en la manipulación de moléculas de ADN que resuelven instancias concretas de distintos problemas combinatorios, algunos computacionalmente intratables.

En 1995 aparecen los primeros modelos de computación ADN, basados en la instanciación de una serie de operaciones bioquímicas consideradas como básicas. En 1996, D. Beaver ([4]) demuestra que todos esos modelos de computación ADN son universales, en el sentido de que todo aquello que es "computable" por una máquina de Turing lo es, así mismo, por una máquina ADN.

2 Dos modelos de computación molecular

En esta sección se describen dos modelos abstractos de computación molecular: *el modelo restringido* de L. Adleman ([2]), y *el modelo débil* de M. Amos ([3]). Los datos de dichos modelos van a ser unos objetos, denominados *tubos*, asociados a un alfabeto arbitrario prefijado, Σ. Los elementos de cada tubo representarán moléculas de ADN, asociando a cada símbolo del alfabeto un oligo verificando ciertas condiciones.

Las instrucciones básicas del modelo serán de dos tipos: *moleculares*, operaciones basadas en propiedades de las moléculas de ADN, y *robóticas*, operaciones estándar (bucles, asignaciones o etiquetados, etc.) que, básicamente, proporcionan la secuenciación de las operaciones moleculares.

Las operaciones moleculares primitivas de los modelos que se introducen son implementables hoy día en el laboratorio con las técnicas actuales de biología molecular. A partir de las instrucciones moleculares y de las convencionales, se definen de manera natural los programas sobre el modelo. Las entradas serán tubos y las salidas serán **SI, NO**, o bien una molécula del tubo final.

Estos modelos de computación molecular están basados en el procedimiento de *filtraje*; es decir, las computaciones en dichos modelos parten de un tubo inicial que contiene todas las posibles soluciones del problema y, mediante sucesivos filtrados, se obtiene un tubo de salida que contiene todas las soluciones correctas del mismo y sólo esas.

En estos modelos, para verificar formalmente un programa molecular diseñado para resolver un problema hay que demostrar dos resultados básicos: (a) toda molécula del tubo de salida representa una solución correcta del problema (*corrección* del programa), y (b) toda molécula del tubo inicial que codifica una solución correcta del problema debe estar en el tubo de salida (*completitud* del programa).

2.1 Modelo restringido de Adleman

Definición: *Un* **agregado** *sobre* Σ *es un multiconjunto finito de elementos de* Σ.
Definición: *Un* **tubo** *sobre* Σ *es un multiconjunto finito de agregados sobre* Σ.
Las instrucciones moleculares básicas del modelo restringido son las siguientess:

- **Extraer**(T, σ): dado un tubo, T, y un símbolo, $\sigma \in \Sigma$, devuelve dos tubos:
$$+(T,\sigma) = \{\gamma \in T : \sigma \in \gamma\} \quad \text{y} \quad -(T,\sigma) = \{\gamma \in T : \sigma \notin \gamma\}$$
- **Mezclar**(T_1, T_2): dados dos tubos, T_1 y T_2, devuelve un nuevo tubo, $T_1 \cup T_2$, que es la unión de ambos, como multiconjuntos.

- **Detectar**(T): dado un tubo, T, devuelve **SI**, en el caso en que T contenga algún agregado, y **NO** en caso contrario.

2.2 Modelo débil de Amos

Definición: *Un tubo sobre* Σ *es un multiconjunto finito de cadenas sobre* Σ.
Las instrucciones moleculares primitivas del modelo débil son las siguientess:

- **Quitar**$(T, \{\gamma_1, ..., \gamma_k\})$: Dado un tubo, T, y un número finito de cadenas, $\gamma_1, ..., \gamma_k$, de Σ, devuelve el tubo obtenido de T eliminando todas aquellas cadenas que contengan, al menos, una ocurrencia de alguna de las cadenas $\gamma_1, ..., \gamma_k$ (téngase presente que $T =$ **quitar**$(T, \emptyset)$).
- **Copiar**$(T, \{T_1, ..., T_k\})$: Dado un tubo, T, devuelve k tubos, $T_1, ..., T_k$, copias exactas de T.
- **Union**$(\{T_1, ..., T_k\}, T)$: Dados los tubos $T_1, ..., T_k$, devuelve un tubo, T, cuyo contenido es la unión de los tubos $T_1, ..., T_k$ como multiconjuntos.
- **Seleccion**(T): Dado un tubo, T, selecciona aleatoriamente un elemento de T en el caso en que $T \neq \emptyset$; en caso contrario, devuelve **NO**.

3 El problema SAT en el modelo restringido

La teoría de la NP–completitud se inicia, propiamente, en 1971 cuando S. Cook establece que el problema **SAT** de la satisfactibilidad de la Lógica Proposicional es NP–duro (es decir, que todo problema de la clase NP es reducible a **SAT** en tiempo polinomial).

En 1995, R. Lipton extendió las ideas de Adleman para resolver cualquier instancia del problema **SAT**, proporcionando un experimento que resolvió una instancia concreta del mismo. En esta sección vamos a formalizar computacionalmente el experimento de Lipton en el modelo restringido.

Recordemos el enunciado del problema SAT: *Dada una fórmula proposicional en forma normal conjuntiva, decidir si existe una asignación de verdad de las variables que haga verdadera dicha fórmula.*
Sea $\varphi \equiv c_1 \wedge ... \wedge c_p$, con $c_i = l_{i,1} \vee ... \vee l_{i,r_i}$ una fórmula proposicional en forma normal conjuntiva, cuyo conjunto de variables es $\mathrm{Var}(\varphi) = \{x_1, ..., x_n\}$. Asociamos a φ el grafo dirigido, $G_\varphi = (V_\varphi, E_\varphi)$, definido así: $V_\varphi = \{a_i, x_i^j, a_{n+1} : \ 1 \leq i \leq n, \ 0 \leq j \leq 1\}$, $E_\varphi = \{(a_i, x_i^j), (x_i^j, a_{i+1}) : \ 1 \leq i \leq n, \ 0 \leq j \leq 1\}$

El grafo G_φ verifica que existen 2^n caminos simples desde a_1 hasta a_{n+1}, así como una biyección natural entre el conjunto de dichos caminos y las valoraciones relevantes para φ.

3.1 Diseño del programa molecular

Consideremos el alfabeto $\Sigma = \{(a_1, x_1^{j_1}, a_2, x_2^{j_2}, ..., x_n^{j_n}, a_{n+1}) : \ \forall i \ (1 \leq i \leq n \rightarrow j_i = 0 \vee j_i = 1)\}$.

El tubo de entrada es $T_0 = \{\sigma \subseteq \Sigma : \ \sigma$ es un conjunto unitario$\}$. Obsérvese que cada elemento de T_0 está identificado por una molécula que codifica una valoración

relevante para la fórmula. El tubo inicial depende únicamente del número de variables de la fórmula φ, por lo que todas las fórmulas que tengan el mismo número de variables podrán usarlo como tubo de partida.

La idea del experimento de Lipton es la siguientes: a partir del tubo inicial se seleccionan todas las moléculas que codifican valoraciones que hacen verdadera la primera cláusula; de éstas se seleccionan las que hacen verdadera la segunda cláusula, y así sucesivamente. En cada paso del proceso anterior, para elegir las moléculas que hacen verdadera una cláusula, se seleccionan las que hacen verdadero el primer literal; de las que hacen falso dicho literal se seleccionan las que hacen verdadero el segundo literal, y así sucesivamente.
Estas ideas sugieren el diseño del siguientes programa molecular:

```
Entrada:  T_0 (en las condiciones anteriores)
          Para i ← 1 hasta p hacer
               T_{i,0} ← ∅;   T''_{i,0} ← T_{i-1}
               Para j ← 1 hasta r_i hacer
                    T'_{i,j} ← +(T''_{i,j-1}, l^1_{i,j})
                    T''_{i,j} ← −(T''_{i,j-1}, l^1_{i,j})
                    T_{i,j} ← T_{i,j-1} ∪ T'_{i,j}
               T_i ← T_{i,r_i}
          Detectar(T_p)
```

En donde, para cada literal, $l_{i,j}$, que aparece en φ, si $l_{i,j} = x_m$, notaremos $l^1_{i,j} = x^1_m$, $l^0_{i,j} = x^0_m$, y si $l_{i,j} = \overline{x}_m$, entonces notaremos $l^1_{i,j} = x^0_m$, $l^0_{i,j} = x^1_m$.

El coste en tiempo molecular (número de operaciones moleculares) del programa, así como el número de tubos usados, es lineal en el número de literales de la fórmula de entrada.

3.2 Verificación formal del programa molecular

Para establecer la verificación formal del programa anterior respecto del problema **SAT**, consideramos las siguientess fórmulas:

- Para cada i $(1 \leq i \leq p)$, se define $\varphi_i \equiv c_1 \wedge ... \wedge c_i$. Sea φ_0 una tautología.
- Para cada i, j tales que $1 \leq i \leq p,\ 1 \leq j \leq r_i$, se define $L_{i,j} \equiv l_{i,1} \vee ... \vee l_{i,j}$.
- Para cada i, j tales que $1 \leq i \leq p,\ 1 \leq j \leq r_i$, se define

  $$\psi(i,j) \equiv \forall \sigma \in T_{i,j}\ (\sigma(\varphi_{i-1} \wedge L_{i,j}) = 1) \wedge \forall \sigma \in T''_{i,j}\ (\sigma(\varphi_{i-1}) = 1 \wedge \sigma(L_{i,j}) = 0)$$

- Para cada i $(1 \leq i \leq p)$, se define $\theta(i) \equiv \forall j\ (1 \leq j \leq r_i \rightarrow \psi(i,j))$.

Teorema 1: La fórmula θ es un invariante del bucle principal. Es decir, $\forall i\ (1 \leq i \leq p \rightarrow \theta(i))$.
Demostración: Por inducción sobre i.
En primer lugar veamos que se verifica $\forall j\ (1 \leq j \leq r_1 \rightarrow \psi(1,j))$, por inducción sobre j.

• La fórmula $\psi(1,1)$ es verdadera, ya que para cada $\sigma \in T_0$ se tiene que

$$\sigma \in T_{1,1} \Longrightarrow \sigma \in T'_{1,1} = +(T_0, l^1_{1,1}) \Longrightarrow \sigma \in T_0 \wedge \sigma(l_{1,1}) = 1 \Longrightarrow \sigma(L_{1,1}) = 1$$
$$\sigma \in T''_{1,1} \Longrightarrow \sigma \in -(T_0, l^1_{1,1}) \Longrightarrow \sigma \in T_0 \wedge \sigma(l_{1,1}) = 0 \Longrightarrow \sigma(L_{1,1}) = 0$$

• Sea $j < r_1$ tal que la fórmula $\psi(1,j)$ es verdadera. Se tiene que

$$\begin{aligned}\sigma \in T_{1,j+1} &\Longrightarrow \sigma \in T_{1,j} \vee \sigma \in T'_{1,j+1} \Longrightarrow (\sigma(L_{1,j}) = 1) \vee (\sigma \in T''_{1,j} \ \wedge \ \sigma(l_{1,j+1}) = 1) \\ &\Longrightarrow \sigma(L_{1,j+1}) = 1 \\ \sigma \in T''_{1,j+1} &\Longrightarrow \sigma \in T''_{1,j} \ \wedge \ \sigma(l_{1,j+1}) = 0 \Longrightarrow \sigma(L_{1,j}) = 0 = \sigma(l_{1,j+1}) \\ &\Longrightarrow \sigma(L_{1,j+1}) = 0\end{aligned}$$

Sea $i < p$ tal que $\theta(i)$ es verdadera. Veamos que $\forall j \ (1 \leq j \leq r_{i+1} \rightarrow \psi(i+1,j))$, por inducción sobre j.

• La fórmula $\psi(i+1,1)$ es verdadera, ya que

$$\sigma \in T_{i+1,1} \Longrightarrow \sigma \in T_i \wedge \sigma(l_{i+1,1}) = 1 \Longrightarrow \sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,1}) = 1$$
$$\sigma \in T''_{i+1,1} \Longrightarrow \sigma \in T_i \wedge \sigma(l_{i+1,1}) = 0 \Longrightarrow \sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,1}) = 0$$

• Sea $j < r_{i+1}$ tal que la fórmula $\psi(i+1,j)$ es verdadera. Se tiene que

$$\begin{aligned}\sigma \in T_{i+1,j+1} &\Longrightarrow \sigma \in T_{i+1,j} \vee \sigma \in T'_{i+1,j+1} \\ &\Longrightarrow (\sigma(\varphi_i \wedge L_{i+1,j}) = 1) \vee (\sigma \in T''_{i+1,j} \ \wedge \ \sigma(l_{i+1,j+1}) = 1) \\ &\Longrightarrow (\sigma(\varphi_i \wedge L_{i+1,j+1}) = 1) \vee (\sigma(\varphi_i) = 1 \ \wedge \sigma(L_{i+1,j}) = 0 \ \wedge \ \sigma(l_{i+1,j+1}) = 1) \\ &\Longrightarrow \sigma(\varphi_i \wedge L_{i+1,j+1}) = 1 \\ \sigma \in T''_{i+1,j+1} &\Longrightarrow \sigma \in T''_{i+1,j} \wedge \sigma(l_{i+1,j+1}) = 0 \\ &\Longrightarrow \sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,j}) = 0 \wedge \sigma(l_{i+1,j+1}) = 0 \\ &\Longrightarrow \sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,j+1}) = 0\end{aligned}$$

□

Corolario 2 (Corrección del programa): $\forall \sigma \in T_p \ (\sigma(\varphi) = 1)$.
Demostración: Se tiene que $T_p = T_{p,r_p}$ y que la fórmula $\psi(p, r_p)$ es verdadera. En consecuencia, para cada $\sigma \in T_p = T_{p,r_p}$ se tiene que $\sigma(\varphi) = \sigma(\varphi_{p-1} \wedge L_{p,r_p}) = 1$. □
Teorema 3: Sea $\sigma \in T_0$ tal que $\sigma(c_1 \wedge ... \wedge c_p) = 1$. Entonces $\forall i \ (1 \leq i \leq p \rightarrow \sigma \in T_i)$.
Demostración: Sea $\sigma \in T_0$ tal que $\sigma(c_1 \wedge ... \wedge c_p) = 1$. Entonces para cada i $(1 \leq i \leq p)$ existe j $(1 \leq j \leq r_i)$ tal que $\sigma(l_{i,j}) = 1$. Notemos $m_i = \text{mín}\{j : 1 \leq j \leq r_i \wedge \sigma(l_{i,j}) = 1\}$. Probemos por inducción sobre i que $\forall i \ (1 \leq i \leq p \rightarrow \sigma \in T_i)$.

Por definición se tiene que $\forall j \ (1 \leq j < m_1 \rightarrow \sigma(l_{1,j}) = 0)$. Como $\sigma \in T_0 = T''_{1,0}$ resulta que $\forall j \ (1 \leq j < m_1 \rightarrow \sigma \in T''_{1,j})$. Luego $\sigma \in T''_{1,m_1-1}$. Teniendo presente que $\sigma(l_{1,m_1}) = 1$, se deduce que $\sigma \in +(T''_{1,m_1-1}, l^1_{1,m_1}) = T'_{1,m_1} \subseteq T_{1,r_1} = T_1$.

La prueba en el paso inductivo es análoga al caso base, usando la hipótesis de inducción.

□

Corolario 4 (Completitud del programa): $\forall \sigma \in T_0 \ (\sigma(\varphi) = 1 \rightarrow \sigma \in T_p)$.

4 EL problema 3-COL en el modelo débil

En esta sección vamos a diseñar una programa molecular en el modelo débil que resuelve el problema **3–COL** (que es un problema **NP**-completo) y estableceremos la verificación formal del mismo.

Definición: *Sean $G = (V, E)$, un grafo no dirigido y $k \geq 1$. Una coloración de G con k colores es una aplicación de V en el conjunto $\{1, ..., k\}$. Diremos que una coloración, f, de G con k colores es válida si y sólo si $\forall u \ \forall v \ (\{u, v\} \in E \Rightarrow f(u) \neq f(v))$.*

Problema 3-COL: *Dado un grafo no dirigido, determinar si posee una coloración válida con tres colores.*

Sean $p_1, ..., p_n$ códigos moleculares ADN de los nodos $1, ..., n$, respectivamente, de un grafo no dirigido $G = (V, E)$. Sean c_1, c_2, c_3 códigos moleculares ADN de los colores $1, 2$ y 3 respectivamente. Para cada i, j $(1 \leq i \leq n, 1 \leq j \leq 3)$ notaremos el par ordenado (p_i, c_j) por $p_i(c_j)$, y representará que el nodo codificado por p_i está coloreado con el color codificado por c_j.

4.1 Diseño del programa molecular

Consideremos el siguientes alfabeto: $\Sigma = \{(p_i, c_j) : 1 \leq i \leq n \wedge 1 \leq j \leq 3\}$.

El tubo de entrada es $T = \{\sigma \in \Sigma^n : \exists x_1 ... \exists x_n (\sigma = (p_1, x_1)(p_2, x_2)...(p_n, x_n))\}$. Es decir, dicho tubo contiene moléculas que codifican cualquier posible coloración del grafo. Si $\sigma \in T$, notaremos $(\sigma)_r = x_r$, para cada r $(1 \leq r \leq n)$.

La idea para resolver el problema **3–COL** es la siguientes: del tubo de entrada se seleccionan las moléculas que codifican coloraciones válidas con tres colores del subgrafo inducido por las aristas cuyo extremo inferior es ≤ 1; de éstas se seleccionan las que codifican coloraciones válidas con tres colores del subgrafo inducido por las aristas cuyo extremo inferior es ≤ 2, y así sucesivamente.

Estas ideas sugieren el diseño del siguientes programa molecular:

```
Entrada:  T (en las condiciones antes citadas)
          Para i ← 1 hasta n − 1 hacer
               copiar(T, {T1, T2, T3})
               Para j ← 1 hasta 3 hacer
                    quitar(Tj, {xi ≠ j, pk(cj) :  k > i ∧ {i, k} ∈ E})
               union({T1, T2, T3}, T)
          Seleccion(T)
```

El coste en tiempo molecular del programa, así como el número total de tubos usados, es lineal en el tamaño del grafo.

4.2 Verificación formal del programa molecular

A fin de establecer la verificación formal del programa diseñado, etiquetaremos los tubos que se obtienen a lo largo de la ejecución, lo cual nos permitirá identificarlos en cualquier instante del proceso.

```
Entrada:  T^0 (en las condiciones antes citadas)
          Para i ← 1 hasta n − 1 hacer
               copiar(T^{i−1}, {T_1^{i−1}, T_2^{i−1}, T_3^{i−1}})
               Para j ← 1 hasta 3 hacer
                    T̄_j^i ← quitar(T_j^{i−1}, {x_i ≠ j, p_k(c_j) :  k > i ∧ {i,k} ∈ E})
               union({T̄_1^i, T̄_2^i, T̄_3^i}, T^i)
          Seleccion(T^{n−1})
```

Del etiquetado de tubos que ha sido introducido se deducen las consideraciones siguientess:

(a) $\forall i\ (1 \leq i \leq n-1 \rightarrow T^i \subseteq T^{i-1})$. En efecto: para cada i, j $(1 \leq i \leq n-1, 1 \leq j \leq 3)$ se tiene que $\overline{T}^i_j \subseteq T^{i-1}_j = T^{i-1}$. Luego $T^i = \bigcup_{j=1}^{3} \overline{T}^i_j \subseteq T^{i-1}$.

(b) Para cada i, j $(1 \leq i \leq n-1, 1 \leq j \leq 3)$ se tiene que $\forall \sigma \in \overline{T}^i_j\ ((\sigma)_i = j \wedge \forall k\ (i < k \wedge \{i,k\} \in E \rightarrow (\sigma)_k \neq j))$. Basta tener presente que $\overline{T}^i_j = \mathbf{quitar}(T^{i-1}_j, \{x_i \neq j, p_k(c_j) : k > i \wedge \{i,k\} \in E\})$.

A fin de establecer la verificación formal del programa, para cada i $(1 \leq i \leq n-1)$, se considera la fórmula $\theta(i) = \forall \sigma \in T^i\ \forall r\ \forall s\ (1 \leq r \leq i \wedge r < s \wedge \{r,s\} \in E \rightarrow (\sigma)_r \neq (\sigma)_s)$.

Teorema 5: La fórmula θ es un invariante del bucle principal. Es decir, $\forall i\ (1 \leq i \leq n-1 \rightarrow \theta(i))$

Demostración: Por inducción sobre i.

Se verifica la fórmula $\theta(1)$, ya que

$\sigma \in T^1 \wedge 1 < s \wedge \{1,s\} \in E \Longrightarrow \exists j\ (1 \leq j \leq 3 \wedge \sigma \in \overline{T}^1_j) \wedge 1 < s \wedge \{1,s\} \in E \stackrel{(b)}{\Longrightarrow} (\sigma)_1 = j\ \wedge\ (\sigma)_s \neq j$

Sea $i < n-1$ tal que $\theta(i)$ es verdadera. Sean $\sigma \in T^{i+1}$ y r, s tales que $1 \leq r \leq i+1 \wedge r < s \wedge \{r,s\} \in E$. Sea $j \in \{1,2,3\}$ tal que $\sigma \in \overline{T}^{i+1}_j$ (que es único por (b)).

Entonces, o bien $r \leq i$, en cuyo caso teniendo presente que $\sigma \in T^{i+1} \stackrel{(a)}{\subseteq} T^i$, de la hipótesis de inducción se deduce que $(\sigma)_r \neq (\sigma)_s$; o bien que $r = i+1$, en cuyo caso de (b) resulta que $(\sigma)_r = (\sigma)_{i+1} = j \wedge (\sigma)_s \neq j$.

□

Corolario 6 (Corrección del programa): $\forall \sigma \in T^{n-1}$ (σ codifica una coloración válida)

Demostración: Del teorema anterior resulta que es verdadera la fómula:

$\theta(n-1) \equiv \forall \sigma \in T^{n-1}\ \forall r\ \forall s\ (1 \leq r \leq n-1 \wedge r < s \wedge \{r,s\} \in E \rightarrow (\sigma)_r \neq (\sigma)_s)$

□

Teorema 7: Sea $\sigma \in T^0$ tal que $\forall r\ \forall s\ (1 \leq r \leq n-1 \wedge r < s \wedge \{r,s\} \in E \rightarrow (\sigma)_r \neq (\sigma)_s)$. Entonces $\forall i\ (1 \leq i \leq n-1 \rightarrow \sigma \in T^i)$

Demostración: Por inducción sobre i.

Teniendo presente que $\sigma \in T^0 = T^0_1 = T^0_2 = T^0_3$ y que σ codifica una coloración válida del grafo, se deduce que $\sigma \in \mathbf{quitar}(T^0_{(\sigma)_1}, \{x_1 \neq (\sigma)_1, p_k(c_{(\sigma)_1}) : k > 1 \wedge \{1,k\} \in E\})$.

Por tanto, $\sigma \in \overline{T}^1_{(\sigma)_1} \subseteq T^1$.

Sea i $(1 \leq i \leq n-2)$, tal que $\sigma \in T^i$. Como $\sigma \in T^i = T_1^i = T_2^i = T_3^i$ y σ codifica una coloración válida del grafo, resulta que $\sigma \in \mathbf{quitar}(T^i_{(\sigma)_{i+1}}, \{x_i \neq (\sigma)_{i+1}, p_k(c_{(\sigma)_{i+1}}) : k > i+1 \wedge \{i+1, k\} \in E\})$. Por tanto, $\sigma \in \overline{T}^{i+1}_{(\sigma)_{i+1}} \subseteq T^{i+1}$.

□

Corolario 8 (Completitud del programa): Sea $\sigma \in T^0$ tal que codifica una coloración válida del grafo. Entonces $\sigma \in T^{n-1}$.

5 Amortiguación de errores

En la descripción de un modelo de computación, las operaciones primitivas del mismo tienen un comportamiento formal inequívoco. Ahora bien, la implementación práctica de dichas operaciones a través de una serie de reacciones químicas puede dar lugar a la existencia de errores, en el sentido de que el resultado obtenido no coincida exactamente con la evaluación formal de la operación implementada.

En el caso de los modelos restringido y débil, las operaciones que pueden acarrear mayores problemas son **extraer** y **quitar**. Vamos a hacer un pequeño análisis de las dificultades que pueden surgir en la implementación práctica de la operación **extraer**, y exponer un procedimiento que permita, en cierto sentido, atenuar el efecto negativo de los errores cometidos.

En la implementación de la operación **extraer**(T, σ), a través de la manipulación de moléculas de ADN, puede ocurrir que tras la ejecución de la misma, alguna molécula que debería estar en $+(T, \sigma)$ (resp. $-(T, \sigma)$) no se encuentre en dicho tubo. Ello puede dar lugar a que, o bien alguna solución correcta del problema se "pierda" a lo largo de la ejecución, con lo cual es posible que el tubo de salida esté vacío y, en cambio, exista alguna solución correcta del problema; o bien que alguna solución incorrecta aparezca en el tubo de salida (en este caso el error tendría menor trascendencia ya que bastaría añadir a la operación **detectar**(T) una nueva operación que verificase si la molécula que ha permitido responder **SI** es, efectivamente, una solución correcta del problema).

Supongamos que los posibles errores que aparecen en la implementación de la operación **extraer** (T, σ) no dependen del tubo T ni del símbolo σ. Notemos p_+ la probabilidad de que tras la ejecución de **extraer**(T, σ), una molécula que debería estar en $+(T, \sigma)$ no esté en dicho tubo. Así, la probabilidad de que una molécula de T que contiene a σ esté en el tubo $+(T, \sigma)$ es $1 - p_+$.

Un procedimiento para conseguir atenuar el efecto del error citado es el siguientes:

```
Entrada: (T, σ, n)
         T_0 ← ∅;   T'_0 ← T
         Para i ← 1 hasta n hacer
              T_i ← +(T'_{i-1}, σ);  T'_i ← −(T'_{i-1}, σ)
              T^+ ← T_{i-1} ∪ T_i
         T^− ← T'_n
```

La ejecución del programa anterior devuelve dos tubos, T^+ y T^-, de tal manera que la probabilidad de que una molécula de T que contiene a σ esté en el tubo T^+

es $1 - (p_+)^n$. Por tanto, es posible amortiguar el error antes citado a unos niveles prefijados con tal de elegir convenientemente el número de pasos a realizar en el procedimiento antes descrito.

Referencias

[1] Adleman, L. Molecular Computation of Solutions to Combinatorial Problems, *Science*, 268, November 1994, 1021–1024.

[2] Adleman, L. On constructing a molecular computer, in *DNA based computers*, R.J. Lipton and E.B. Baum, eds., American Mathematical Society, 1996, 1–22.

[3] Amos, M.; Wilson, S.; Hodgson, D.A.; Owenson, G.; Gibbons, A. Practical implementation of DNA computations, in *Unconventional Models of Computation*, Springer, 1998, 1–18.

[4] Beaver, D. A universal molecular computer, in *DNA based computers*, R.J. Lipton and E.B. Baum, eds., American Mathematical Society, 1996, 29–36.

[5] Lipton R.J. DNA Solution of Hard Computational Problems, *Science*, 268, April 1995, 542–545.