Soluciones moleculares del problema SAT de la Lógica Proposicional

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

Abstract

The aim of this paper is the computational formalization of Lipton's experiment about **SAT** problem of Propositional Logic. We present two models in molecular computation: *restricted* and *weak* models, due to L. Adleman and M. Amos, respectively. We design a formal molecular program in each one of these models, based in DNA, that solve **SAT** following Lipton's ideas. Using invariant techniques, we prove the soundness and completeness of these molecular programs.

1 Introducción

La resolubilidad algorítmica *práctica* de problemas está relacionada directamente con la potencia de cálculo y la densidad de almacenamiento de información de los ordenadores convencionales.

Hacia finales de la década de los cincuenta, R. Feynman introduce el concepto teórico de *computación a nivel molecular*, postulándolo como una innovación revolucionaria en la carrera por la miniaturización. Las ideas de Feynman adquieren una especial relevancia a partir de 1983, cuando R. Churchhouse establece las limitaciones físicas de la velocidad de cálculo de un ordenador convencional (que representa la información sobre chips de silicio a través de una serie de impulsos eléctricos, interpretados como ceros y unos, y manipula la misma mediante la realización de computaciones con esos impulsos).

El **ADN** (ácido desoxirribonucleico) es una molécula fundamental de los seres vivos. En 1951, J. Watson y F. Crick comienzan a descifrar la estructura de las moléculas de ADN, compuestas básicamente por una sucesión de cuatro bases nucleótidas (adenina, **A**, citosina, **C**, guanina, **G**, y timina, **T**), descubriendo en 1953 el principio de complementariedad (por el que unas cadenas simples de ADN pueden enlazarse con otras, de acuerdo con el principio **T-A**, **C-G**, dando lugar a la famosa estructura de doble hélice). Watson y Crick demuestran que las moléculas de ADN codifican toda la información genética de los organismos vivos (en contra de la creencia generalizada que reservaba a las proteinas esa cualidad fundamental) y justifican cómo es posible utilizar ciertas técnicas para su manipulación.

Al igual que en una CPU convencional existen unas operaciones básicas, aritméticas y lógicas, a partir de las cuales se puede realizar cualquier otra operación, entre moléculas de ADN es posible desarrollar unas operaciones enzimáticas básicas (cortar, pegar, copiar, seleccionar, etc...), que, a su vez, permiten la simulación de cualquier procedimiento mecanizable a través del ADN, con la peculiaridad de que las enzimas trabajan simultáneamente sobre dichas moléculas.

De esta manera surge la posibilidad de implementar algoritmos a través de moléculas de ADN, debido a que éstas se pueden usar como una especie de memoria (representando la información mediante una codificación basada en cuatro nucleótidos) siendo susceptible la manipulación de las mismas (con tal de efectuar con ellas una serie de reacciones químicas diversas).

En noviembre de 1994, L. Adleman ([1]) consigue resolver una instancia concreta del *problema del circuito hamiltoniano* (en su versión dirigida y con un par de nodos distinguidos) a través de la manipulación de moléculas de ADN usando técnicas de biología molecular. Por primera vez se consigue resolver un problema matemático mediante el uso de herramientas bioquímicas, cuando hasta entonces había sido frecuente el procedimiento inverso; es decir, el uso de resultados matemáticos para resolver problemas bioquímicos.

El experimento de Adleman resolvió el problema citado para un grafo concreto de tamaño siete. Dicha instancia puede ser resuelta por una persona en unos minutos y por un ordenador en menos de un segundo, mientras que Adleman tardó siete días ¿Dónde radica la excepcionalidad de dicho experimento? Algunos motivos que justifican la extraordinaria relevancia del mismo son los siguientes:

- Proporciona un primer ejemplo de computación a nivel molecular, que potencialmente es un tamaño que nunca podrá ser alcanzado por la industria de los semiconductores.
- Muestra nuevas perspectivas de las moléculas de ADN como estructura de datos muy singulares (que resultan de la complementariedad de Watson-Crick y de la extraordinaria densidad de información capaz de codificar).
- Ilustra la posibilidad de usar moléculas de ADN para resolver instancias de problemas computacionalmente intratables (recuérdese que el problema citado es **NP**-completo).
- Manifiesta la capacidad del ADN para simular trabajos de forma masivamente paralela, gracias a la acción simultánea de las enzimas sobre dichas moléculas.

Con el experimento de Adleman nace propiamente la computación ADN y se pone de manifiesto las presumibles ventajas de esta computación respecto de los ordenadores electrónicos convencionales, en lo que respecta a velocidad de cálculo, consumo de energía y densidad de información.

En abril de 1995, R.J. Lipton ([5]) resuelve una instancia del *problema* **SAT** *de la satisfactibilidad de la Lógica Proposicional* siguiendo la línea de Adleman, con la peculiaridad de que el tubo de ensayo inicial no depende del dato de entrada concreto (una fórmula proposicional en forma normal conjuntiva), sino únicamente de su "tamaño" (el número de variables de la fórmula).

Así, mediante la realización de sucesivos experimentos basados en la manipulación de moléculas de ADN, se van resolviendo instancias concretas de distintos tipos de problemas combinatorios, muchos de ellos computacionalmente intratables. Ahora bien, ¿qué clase de problemas pueden ser resueltos, al menos teóricamente, mediante la computación ADN?

A principios de 1995 comienzan a aparecer los primeros modelos de computación ADN, basados en la instanciación de una serie de operaciones bioquímicas consideradas como básicas o primitivas en el modelo. En 1996, D. Beaver ([4]) demuestra que todos esos modelos de computación ADN son computacionalmente completos, en el sentido de que todo aquello que es computable por una máquina de Turing lo es, así mismo, por una máquina ADN.

En este trabajo se presentan dos modelos de computación molecular: el *modelo restringido* de L. Adleman ([2]) y el *modelo débil* de M. Amos ([3]). Se estudian programas en dichos modelos que resuelven el problema **SAT**, siguiendo las ideas de Lipton, y se establece la verificación formal de los programas moleculares presentados utilizando la técnica de inducción sobre la estructura de los mismos.

2 Modelo restringido y modelo débil

En esta sección se introducen dos modelos abstractos de computación molecular: *el modelo restringido* de L. Adleman y *el modelo débil* de M. Amos.

Los datos de dichos modelos van a ser unos objetos, denominados *tubos*, asociados a un alfabeto arbitrario prefijado, Σ. Los elementos de cada tubo codificarán moléculas de ADN, con tal de asociar a cada símbolo del alfabeto un oligo (cadena corta sintética de bases nucleótidas) verificando ciertas condiciones.

Las instrucciones básicas de estos modelos serán de dos tipos:

- *Moleculares*: operaciones basadas en propiedades de las moléculas de ADN.
- *Robóticas*: operaciones estándar (bucles, condicionales, asignaciones o etiquetados, etc.) que, básicamente, proporcionan la secuenciación de las operaciones moleculares.

La elección de las operaciones moleculares primitivas de los modelos que se introducen se debe, entre otros motivos, al hecho de que todas ellas son implementables hoy día con las técnicas actuales de biología molecular.

A partir de las instrucciones moleculares y de las convencionales, se obtienen de manera natural los programas sobre el modelo. Las entradas de los mismos serán tubos y las salidas serán **SI, NO**, o bien un elemento (una molécula) del tubo final.

Los modelos restringido y débil son modelos de computación molecular 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. Así pues, las computaciones en dichos modelos no alteran la estructura interna de las moléculas de ADN que integran los tubos, simplemente las moléculas son rechazadas o no de acuerdo con un determinado test, o bien son replicadas.

En estos modelos, para verificar formalmente un programa molecular diseñado para resolver un problema (es decir, para establecer que el programa resuelve *realmente* el problema), hay que demostrar dos resultados básicos:

(a) Toda molécula del tubo de salida representa una solución correcta del problema (*corrección* del programa).

(b) Toda molécula del tubo inicial que codifica una solución correcta del problema debe estar en el tubo de salida (*completitud* del programa); es decir, si el tubo de salida está vacío, entonces no existe ninguna solución correcta del problema.

De acuerdo con las consideraciones anteriores vamos a introducir los modelos restringido y débil definiendo el concepto de tubo y explicitando las operaciones moleculares primitivas de cada uno de ellos.

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 siguientes:

- **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.

Obsérvese que, en principio, el concepto de tubo en este modelo es distinto del dado en el modelo restringido. La diferencia esencial desde el punto de vista molecular radica en la consideración del direccionamiento u orientación de las moléculas que integran el tubo. Obviamente, todo tubo del modelo débil puede considerarse, también, como un tubo del modelo restringido, pero no al contrario.
Las instrucciones moleculares primitivas del modelo débil son las siguientes:

- **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 $T = \text{quitar}(T, \emptyset)$).

- **Copiar**$(T, \{T_1, ..., T_k\})$: Dado un tubo, T, devuelve k tubos, $T_1, ..., T_k$, que son 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 Solución molecular del problema SAT 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 (esto es, decidir si la fórmula es satisfactible).*

Sea φ una fórmula proposicional en forma normal conjuntiva dada por la expresión

$$\varphi \equiv c_1 \wedge ... \wedge c_p, \text{con } c_i = l_{i,1} \vee ... \vee l_{i,r_i}$$

y cuyo conjunto de variables es $\text{Var}(\varphi) = \{x_1, ..., x_n\}$.

Asociamos a la fórmula φ un grafo dirigido, $G_\varphi = (V_\varphi, E_\varphi)$, definido como sigue:

$$V_\varphi = \{a_i, x_i^j, a_{n+1} : \ 1 \leq i \leq n, \ 0 \leq j \leq 1\}$$

$$E_\varphi = \{(a_i, x_i^j), (x_i^j, a_{i+1}) : \ 1 \leq i \leq n, \ 0 \leq j \leq 1\}$$

El grafo G_φ verifica las siguientes propiedades:

- Existen 2^n caminos simples desde a_1 hasta a_{n+1} en G_φ.
- Existe una biyección natural entre el conjunto de los caminos anteriormente citados y las valoraciones relevantes para φ: Si $\gamma = a_1 x_1^{j_1} a_2 x_2^{j_2} ... x_n^{j_n} a_{n+1}$ es un tal camino, entonces se le asocia la valoración, σ, relevante para φ, caracterizada por las relaciones $\sigma(x_i) = j_i$ $(1 \leq i \leq n)$.

Con el grafo G_φ se procede como en el experimento de Adleman, codificando los nodos por oligos de longitud fija, y los arcos por cadenas que dependerán de la codificación elegida para los nodos que conectan. De acuerdo con el método dado en el experimento de Adleman, en el tubo de ensayo inicial se obtiene un multiconjunto de moléculas que codifican todas las valoraciones relevantes para la fórmula φ.

Como hemos indicado anteriormente, a diferencia del experimento de Adleman, este 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 ensayo de partida.

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, T_0, es el siguiente $T_0 = \{\sigma \subseteq \Sigma : \ \sigma \text{ es un conjunto unitario}\}$. Obsérvese que cada elemento de T_0 está identificado por una molécula que codifica una valoración relevante para la fórmula.

Para cada literal, $l_{i,j}$, que aparece en la fórmula φ:

- Si $l_{i,j} = x_m$, entonces notaremos $l_{i,j}^1 = x_m^1$, $l_{i,j}^0 = x_m^0$.
- Si $l_{i,j} = \overline{x}_m$, entonces notaremos $l_{i,j}^1 = x_m^0$, $l_{i,j}^0 = x_m^1$.

Es decir, si una molécula contiene $l^1_{i,j}$ (resp. $l^0_{i,j}$), entonces el literal $l_{i,j}$ tiene asignado el valor 1 (resp. 0) por la valoración que codifica dicha molécula.

La idea del experimento de Lipton es la siguiente: a partir del tubo inicial, T_0, que codifica todas las valoraciones relevantes para la fórmula, se procede como sigue.

- Formamos un nuevo tubo, T_1, seleccionando de T_0 todas las valoraciones que hacen verdadera la cláusula c_1. Para ello:
  - ⋆ Se eligen las que hacen verdadero el literal $l_{1,1}$.
  - ⋆ De las que hacen falso $l_{1,1}$, se eligen las que hacen verdadero $l_{1,2}$.
  - ⋆ De las que hacen falso $l_{1,1} \vee l_{1,2}$, se eligen las que hacen verdadero $l_{1,3}$.
  - ⋆ Y así sucesivamente con todos los literales de c_1.
- Formamos un nuevo tubo, T_2, seleccionando de T_1 todas las valoraciones que hacen verdadera la cláusula c_2 (así dichas valoraciones hacen verdadera la fórmula $c_1 \wedge c_2$). Para ello:
  - ⋆ Se eligen las que hacen verdadero el literal $l_{2,1}$.
  - ⋆ De las que hacen falso $l_{2,1}$, se eligen las que hacen verdadero $l_{2,2}$.
  - ⋆ De las que hacen falso $l_{2,1} \vee l_{2,2}$, se eligen las que hacen verdadero $l_{2,3}$.
  - ⋆ Y así sucesivamente con todos los literales de c_2.
- El proceso se reitera p veces hasta obtener el tubo T_p, a partir de T_{p-1}, que contiene todas las valoraciones que hacen verdadera la fórmula $c_1 \wedge \ldots \wedge c_p$.

Es decir, el experimento consta de p pasos estructurados en un bucle FOR, siendo p el número de cláusulas de la fórmula, de tal manera que en la vuelta i–ésima de dicho bucle se genera un tubo, T_i, que contiene únicamente las moléculas que codifican valoraciones que hacen verdadera la fórmula $c_1 \wedge \ldots \wedge c_i$. Para ello se considera un bucle secundario FOR que da r_i vueltas (en donde r_i es el número de literales de la cláusula c_i), de tal manera que la vuelta j–ésima de ese bucle genera dos tubos, $T_{i,j}$ y $T''_{i,j}$, que contienen únicamente las valoraciones que hacen verdadera todas las cláusulas $c_1, \ldots, c_{i-1}$ y, además,

(a) hacen *verdadero* alguno de los literales $l_{i,1}, \ldots, l_{i,j}$ de c_i (en el caso del tubo $T_{i,j}$).

(b) hacen *falso* todos los literales $l_{i,1}, \ldots, l_{i,j}$ de c_i (en el caso del tubo $T''_{i,j}$).

El tubo T_i se define como un nuevo etiquetado de T_{i,r_i}.
Estas ideas sugieren el diseño del siguiente 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)
```

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

$$1 + \sum_{i=1}^{p} (3 + 3 \cdot r_i) = 1 + 3p + 3k \in O(k)$$

3.2 Verificación formal del programa molecular

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

$\boxed{i = 1}$

Veamos que se verifica $\forall j\ (1 \leq j \leq r_1 \rightarrow \psi(1,j))$, por inducción sobre j:

$\boxed{j = 1}$

Se tiene que

$$\begin{array}{l} T'_{1,1} = +(T''_{1,0}, l^1_{1,1}) = +(T_0, l^1_{1,1})\ ,\ \text{ya que } T''_{1,0} = T_0 \\ T''_{1,1} = -(T''_{1,0}, l^1_{1,1}) = -(T_0, l^1_{1,1}) \\ T_{1,1} = T_{1,0} \cup T'_{1,1} = T'_{1,1} = +(T_0, l^1_{1,1})\ ,\ \text{ya que } T_{1,0} = \emptyset \end{array}$$

Como $\sigma \in T_0$ se verifica:

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

Es decir, la fórmula $\psi(1,1)$ es verdadera.

$\boxed{j < r_1 \rightarrow j + 1}$

Supongamos que la fórmula $\psi(1,j)$ es verdadera; es decir, que

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

Se tiene que

$$\begin{array}{l} T'_{1,j+1} = +(T''_{1,j}, l^1_{1,j+1}) \\ T''_{1,j+1} = -(T''_{1,j}, l^1_{1,j+1}) \\ T_{1,j+1} = T_{1,j} \cup T'_{1,j+1} \end{array}$$

Sea $\sigma \in T_{1,j+1}$. Entonces

* O bien $\sigma \in T_{1,j}$, en cuyo caso $\sigma(L_{1,j}) = 1$. Luego $\sigma(L_{1,j+1}) = 1$.
* O bien $\sigma \in T'_{1,j+1} = +(T''_{1,j}, l^1_{1,j+1})$, en cuyo caso $\sigma \in T''_{1,j} \wedge \sigma(l_{1,j+1}) = 1$. Por tanto $\sigma(L_{1,j+1}) = 1$.

Sea $\sigma \in T''_{1,j+1}$. Entonces $\sigma \in T''_{1,j} \ \wedge \ \sigma(l_{1,j+1}) = 0$. Luego $\sigma(L_{1,j}) = 0 = \sigma(l_{1,j+1})$. Es decir, $\sigma(L_{1,j+1}) = 0$.
En consecuencia, la fórmula $\psi(1, j+1)$ es verdadera.

Por tanto, se verifica $\theta(1)$.

$\boxed{i < p \rightarrow i+1}$

Supongamos que la fórmula $\theta(i)$ es verdadera; es decir, que $\forall j \ (1 \leq j \leq r_i \rightarrow \psi(i,j))$. Para cada $j \ (1 \leq j \leq r_i)$, se tiene que

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

Veamos que se verifica $\forall j \ (1 \leq j \leq r_{i+1} \rightarrow \psi(i+1, j))$, por inducción sobre j.

$\boxed{j = 1}$

Se tiene que

$$\begin{array}{l} T'_{i+1,1} = +(T''_{i+1,0}, l^1_{i+1,1}) = +(T_i, l^1_{i+1,1}) \ , \text{ ya que } T''_{i+1,0} = T_i \\ T''_{i+1,1} = -(T''_{i+1,0}, l^1_{i+1,1}) = -(T_i, l^1_{i+1,1}) \\ T_{i+1,1} = T_{i+1,0} \cup T'_{i+1,1} = T'_{i+1,1} = +(T_i, l^1_{i+1,1}) \ , \text{ ya que } T_{i+1,0} = \emptyset \end{array}$$

Sea $\sigma \in T_{i+1,1}$. Entonces $\sigma \in T_i \wedge \sigma(l_{i+1,1}) = 1$. Como $\sigma \in T_i = T_{i,r_i}$, de la hipótesis de inducción se deduce que $\sigma(\varphi_i) = 1$. Por tanto, $\sigma(\varphi_i \wedge L_{i+1,1}) = 1$.
Sea $\sigma \in T''_{i+1,1} = -(T_i, l^1_{i+1,1})$. Entonces $\sigma \in T_i \wedge \sigma(l_{i+1,1}) = 0$. Luego $\sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,1}) = 0$.
En consecuencia, la fórmula $\psi(i+1, 1)$ es verdadera.

$\boxed{j < r_{i+1} \rightarrow j+1}$

Sea $j < r_{i+1}$, tal que la fórmula $\psi(i+1, j)$ es verdadera; es decir, que

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

Se tiene que

$$\begin{array}{l} T'_{i+1,j+1} = +(T''_{i+1,j}, l^1_{i+1,j+1}) \\ T''_{i+1,j+1} = -(T''_{i+1,j}, l^1_{i+1,j+1}) \\ T_{i+1,j+1} = T_{i+1,j} \cup T'_{i+1,j+1} \end{array}$$

Sea $\sigma \in T_{i+1,j+1}$. Entonces

* O bien $\sigma \in T_{i+1,j}$, en cuyo caso $\sigma(\varphi_i \wedge L_{i+1,j}) = 1$. Luego $\sigma(\varphi_i \wedge L_{i+1,j+1}) = 1$.
* O bien $\sigma \in T'_{i+1,j+1} = +(T''_{i+1,j}, l^1_{i+1,j+1})$, en cuyo caso $\sigma \in T''_{i+1,j} \ \wedge \sigma(l_{i+1,j+1}) = 1$. Luego $\sigma(\varphi_i) = 1 \ \wedge \sigma(L_{i+1,j}) = 0 \ \wedge \ \sigma(l_{i+1,j+1}) = 1$; es decir $\sigma(\varphi_i \wedge L_{i+1,j+1}) = 1$.

Sea $\sigma \in T''_{i+1,j+1} = -(T''_{i+1,j}, l^1_{i+1,j+1})$. Entonces $\sigma \in T''_{i+1,j} \wedge \sigma(l_{i+1,j+1}) = 0$. Luego $\sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,j}) = 0 \wedge \sigma(l_{i+1,j+1}) = 0$. Es decir, $\sigma(\varphi_i) = 1 \wedge \sigma(L_{i+1,j+1}) = 0$.
En consecuencia, la fórmula $\psi(i+1, j+1)$ es verdadera.

Esto concluye la prueba del teorema 1. □

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 $\sigma(\varphi_{p-1} \wedge L_{p,r_p}) = 1$. □

Acabamos de probar que toda molécula del tubo de salida codifica una asignación de verdad que hace satisfactible la fórmula. Veamos finalmente que si el tubo de salida es vacío, entonces la fórmula no es satisfactible; es decir, veamos que toda molécula del tubo inicial que codifique una valoración que hace verdadera la fórmula, supera todos los filtros del programa y permanece en el tubo de salida.

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

$$\forall i \ (1 \leq i \leq p \rightarrow \exists j \ (1 \leq j \leq r_i \wedge \sigma(l_{i,j}) = 1))$$

Para cada $i \ (1 \leq i \leq p)$ notemos $m_i = \min\{j : 1 \leq j \leq r_i \wedge \sigma(l_{i,j}) = 1\}$. Probemos por inducción sobre i que $\forall i \ (1 \leq i \leq p \rightarrow \sigma \in T_i)$.

$\boxed{i = 1}$

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 $\sigma \in +(T''_{1,m_1-1}, l^1_{1,m_1}) = T'_{1,m_1} \subseteq T_{1,r_1} = T_1$.

$\boxed{i < p \rightarrow i + 1}$

La prueba 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 Solución molecular del problema SAT en el modelo débil

En esta sección se va a describir un programa molecular en el modelo débil que resuelve el problema **SAT**. Se justificará que para *este* problema es posible *adaptar* al modelo débil el programa molecular diseñado anteriormente para resolver **SAT** en el modelo restringido.
Consideremos el alfabeto $\Sigma = \{a_i, x^j_i, a_{n+1} : \ 1 \leq i \leq n \ \wedge \ 0 \leq j \leq 1\}$.
El tubo de entrada, T_0, es el siguiente conjunto de cadenas de Σ de longitud $2n + 1$:

$$\{\sigma : \ \exists j_1 \ldots \exists j_n \ (\forall i \ (1 \leq i \leq n \rightarrow 0 \leq j_i \leq 1) \ \wedge \ \sigma = a_1 x_1^{j_1} a_2 x_2^{j_2} ... a_n x_n^{j_n} a_{n+1})\}$$

Obsérvese que cada molécula de T_0 codifica una valoración relevante para φ.

Veamos que para el problema **SAT** es posible simular la instrucción *extraer* del modelo restringido a través de la instrucción *quitar* del modelo débil.

En efecto: es obvio que la "operación" $T_1 \leftarrow -(T_2, \gamma)$ del modelo restringido es equivalente a la instrucción $T_1 \leftarrow quitar(T_2, \{\gamma\})$ del modelo débil. Ahora bien, la "operación" $T_1 \leftarrow +(T_2, \gamma)$ del modelo restringido no se puede simular, en general, mediante una instrucción *quitar* del modelo débil. No obstante, en el caso particular del programa molecular diseñado en el modelo restringido, la "operación" $T_1 \leftarrow +(T_2, \gamma)$ tiene la forma $T_1 \leftarrow +(T_2, l^1_{i,j})$, y es evidente que dicha operación puede ser simulada por la instrucción $T_1 \leftarrow quitar(T_2, \{l^0_{i,j}\})$ del modelo débil.

De lo que antecede resulta que el programa molecular en el modelo restringido descrito en la sección 3.1. proporciona el siguiente programa molecular en el modelo débil.

```
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
                   copiar(T''_{i,j-1}, {T^0_{i,j-1}, T^1_{i,j-1}})
                   T'_{i,j} ← quitar(T^0_{i,j-1}, {l^0_{i,j}})
                   T''_{i,j} ← quitar(T^1_{i,j-1}, {l^1_{i,j}})
                   union({T_{i,j-1}, T'_{i,j}}, T_{i,j})
              T_i ← T_{i,r_i}
         Seleccion(T_p)
```

Obsérvese que mientras la salida del programa diseñado en el modelo restringido es **SI** o **NO**, según la satisfactibilidad de la fórmula, este programa devuelve una molécula (que codifica una valoración que hace verdadera la fórmula), o bien, en caso contrario, devuelve **NO**.

La verificación formal de este programa molecular, diseñado para resolver el problema **SAT**, se deduce de las consideraciones realizadas anteriormente, así como de la verificación formal del programa en el modelo restringido que se ha establecido en la sección 3.2.

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

[4] 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.

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