Formal Verification of Programs in Molecular Models with Random Access Memory

MARIO J. PÉREZ-JIMÉNEZ
FERNANDO SANCHO-CAPARRINI

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla, E.T.S. Ingeniería Informática
Avda. Reina Mercedes, s/n – 41012 Sevilla, Spain
E-mail: {Mario.Perez,Fernando.Sancho}@cs.us.es

Abstract

Formal verification of molecular programs is a first step towards their automatic processing by means of reasoning systems (ACL2, PVS, etc). In this paper a systematic method to establish verifications of these programs within molecular models with memory, that is, molecular computing models where some operations modifying the inner structure of molecules exist, is proposed. The method presented in this work is applied to relevant problems for the design of molecular programs solving some well-known numerical **NP**-complete problems: the *Generating Cover Families* problem, the *Set Covering* problem and the *Minimal Set Cover* Selection Problem.

M. J. Pérez-Jiménez et al. (Eds.): Recent Results in Natural Computing, pp. 205–229

1 Introduction

Since the first result in molecular computing at the end of 1994 [1], some solutions to several **NP**-complete problems in this framework have been offered. In 1995 the first molecular models appeared. They are universal models, that is, with the same computational power as Turing Machines. Since then, it appears the possibility of designing molecular programs having as input the instance of the problem to solve (until then, molecular computing was limited to solve particular instances, with no description of molecular operations in general cases). However, this brings about the necessity of a greater flexibility in the design of the programs and the necessity of a formal verification of the fact that those programs effectively solve the problem for which they were designed.

Usually, the first step to solve a concrete problem in computing models is to encode the input of the problem into the type of data that the model deals with (in this case, encoding will be made by means of tubes containing DNA molecules); next step consists in applying a finite sequence of basic operations of the model to the input data to get an output encoding the solution of the problem.

The goal of this paper is the presentation of a methodology to establish a formal verification of programs in molecular models with memory. The method is applied to molecular programs solving several well-known numerical **NP**-complete problems: the *Generating Cover Families* problem, the *Set Covering* problem, and the *Minimal Set Cover* problem.

The problems we solve in this work are associated with a finite family, $\mathcal{F}$, of subsets of a finite set. And they are the following ones: (1) generate every ordered pair $(\mathcal{F}', B)$ such that $\mathcal{F}'$ is a subfamily of $\mathcal{F}$ and $B = \cup\mathcal{F}'$; (2) generate all pairwise disjoint subfamilies of $\mathcal{F}$.

The paper is organized as follows: in section 2 a methodology to establish a formal verification of programs in molecular models with memory is given. Section 3 summarizes the main characteristics of the molecular model we use in this work: the Sticker Model. Sections 4 and 5 apply this methodology to two numerical problems. The main reason why we have chosen the above problems is because there exists a qualitative difference in the way their formal verification have been established. On one hand, all the molecules of the input tube are kept

(possibly modified) along the execution of the program; on the other hand, a filtering process is made, hence some molecules are thrown away because they do not encode any of the valid solutions to the problem.

2 A Methodology for Verification of Programs within Models with Memory

Let P be a program within a molecular model designed to solve a problem X. Let us suppose that the program P has a main loop FOR or $WHILE$ (we will say that P is a FOR or $WHILE$ program). To verify (X, P) we will search formulas, in the variables of the program, that are invariant of the main loop and such that, at the end of the execution, can help us to deduce the soundness and completeness of P for the problem X.

The execution of the program P can be interpreted as the evolution over time of a certain population. At the beginning of the execution the population consists of a multiset of molecules forming the input tube of P. Every molecule represents an element, thus, in the original population cloned members can exist. Every step of the main loop takes one unit of time. Hence, throughout the execution an element of the population can die (that is, it can be thrown away) or it can survive. If the molecular computing model has random access memory (that is, if the inner structure of the molecules can be modified) then the molecules surviving all the steps can be of two kinds: those whose inner structure has been altered and those that remain with no modifications.

The study of invariant formulas of the main loop needs a detailed study of every molecule in the input tube. First we proceed to a relabelling of the tubes used in the program in order to individualize them and to distinguish and characterize the relevant tubes along the execution. Second, a function showing the evolution of every molecule, σ, in every step, i, of the main loop is defined. This function can be undefined for some values (σ, i). This will be interpreted as the fact that the molecule σ does not survive after the execution of i-th step of the loop.

The methodology we propose to verify (X, P) consists of the following stages:

1. Re-labelling of the tubes of the program P, to individualize them along the execution.

2. A detailed follow-up of the evolution of every molecule along the process, using a function that we will name $STEP$.

3. Searching invariant formulas of the main loop based on the $STEP$ function properties.

4. Deducing the *Soundness* (every molecule in the output tube encodes a valid solution of the problem) and the *Completeness* (every molecule in the input tube encoding a valid solution of the problem must be in the output tube) of the program based on invariant formulas after the execution of the program.

3 The Sticker Model

The *sticker model* was introduced by S. Roweis, E. Winfree et al [6] as an abstract model of molecular computing based on DNA, with random access memory. In this model the information is represented in a different way from that used in the Adleman-Lipton paradigm. An (n, k, m)-memory strand, with $n \geq k \cdot m$, consists of a strand of n bases subdivided into k non-overlapping substrands each of which is m bases long. A sticker associated with an (n, k, m)-memory strand is m bases long and complementary to exactly one of the k substrands of the memory strand. If a sticker is annealed to its matching substrand on a memory strand, then the particular substrand is said to be *on*, if not it is said to be *off*. An (n, k, m)-memory complex is an (n, k, m)-memory strand along with its annealed stickers (if any). The (n, k, m)-memory complexes represent bit strings of $\{0, 1\}^k$, for this reason, it is usual to identify them either as binary functions ($\sigma : \{1, ..., k\} \longrightarrow \{0, 1\}$, such that $\sigma(i) = 1$ if and only if the i-th substrand is *on*), or as subsets of $\{1, ..., k\}$ by means of their characteristic functions.

Within the sticker model, a tube is a finite multiset (a collection where elements can be repeated) of memory complexes. In this paper we use the following operations of the sticker model over tubes:

- `Combine`(T_1, T_2): given two test tubes T_1 and T_2 this operation produces a new tube, denoted by $T_1 \cup T_2$, the multiset union of T_1 and T_2.

- `Separate`(T, i): given a test tube T and a natural number i, with $1 \leq i \leq k$, this operation produces two new tubes, $+(T, i) = \{\{\sigma \in T : \sigma(i) = 1\}\}$ and $-(T, i) = \{\{\sigma \in T : \sigma(i) = 0\}\}$.

  We usually write $(T_1, T_2) \leftarrow$ `separate`(T, i) to indicate that $T_1 = +(T, i)$ and $T_2 = -(T, i)$.

- `Set`(T, i): given a test tube T and a natural number i, with $1 \leq i \leq k$, this operation produces a new tube, `Set`(T, i), where the ith substrand of each memory complex in T is turned *on*.

- `Read`(T): given a test tube T this operation reads the content of tube T. To achieve that, one memory complex has to be isolated from T and its annealed stickers determined, or else it has to be reported that the tube T contains no memory complexes.

Note that, in this model, the operations `Separate` and `Set` are the only ones implementing a massive parallelism. Also, only operation `Set` can modify inner structure of the molecules of a tube.

A (k, l)-library, with $1 \leq l \leq k$, consists of all memory complexes with k substrands, where the first l substrands are either *on* or *off*, in all possible ways, whereas the last $k - l$ substrands are *off*.

In the *sticker model* a program, P, is a finite sequence of molecular operations that can be written in a simple way by means of robotic operations (such as loops, conditionals, etc.).

4 The Generating Cover Families Problem

The *Generating Cover Families* problem is the following: *Let* $A = \{1, \ldots, p\}$. *Let* $\mathcal{F} = \{B_1, \ldots, B_q\}$ *be a finite family of subsets of* A. *Determine all ordered pairs* $(\mathcal{F}', B)$, *where* $\mathcal{F}'$ *is a subfamily of* $\mathcal{F}$ *and* $B = \bigcup \mathcal{F}'$.

4.1 Design of a Program in the Sticker Model

To solve this problem we consider as the input tube, T_0, a $(p+q, q)$-library encoding all possible subfamilies of $\mathcal{F}$.

If ρ is a memory complex with $p+q$ substrands (from now on, we will say that ρ is a molecule), we will note:

$$\begin{cases} \rho_q = (\rho(1), \ldots, \rho(q)) \\ \rho_p = (\rho(q+1), \ldots, \rho(q+p)) \end{cases}$$

We can interpret that molecule ρ encodes an ordered pair, $(\mathcal{F}_\rho, A_\rho)$, where $\mathcal{F}_\rho$ is a subfamily of $\mathcal{F}$ and A_ρ is a subset of A, according to the following:

$$\begin{cases} \mathcal{F}_\rho = \{B_k \in \mathcal{F} : \ 1 \leq k \leq q \ \wedge \ \rho(k) = 1\} \\ A_\rho = \{s \in A : \ 1 \leq s \leq p \ \wedge \ \rho(q+s) = 1\} \end{cases}$$

Definition 1. Let ρ be a molecule and let t be such that $1 \leq t \leq q+p$. We will say that $t \in \rho$ (resp. $t \notin \rho$) if and only if $\rho(t) = 1$ (resp. $\rho(t) = 0$).

If ρ is a molecule, then we have:

$$\bigcup \mathcal{F}_\rho = \{x_k^j : \ 1 \leq k \leq q \ \wedge \ 1 \leq j \leq r_k \ \wedge \ k \in \rho\}$$

Definition 2. A molecule ρ is consistent if and only if $A_\rho = \bigcup \mathcal{F}_\rho$.

A molecular program solving the *Generating Cover Families* problem is the following:

```
Procedure Cover
Input:  T_0 (where T_0 is a (p+q,q)-library)
     for i = 1 to q do
          (T_i^+, T_i^-) ← separate(T_0, i)
          for j = 1 to r_i do
               set(T_i^+, q + x_i^j)
          end for
          T_0 ← combine(T_i^+, T_i^-)
     end for
```

Where, for each i $(1 \leq i \leq q)$ we denote by r_i the cardinality of the set B_i, and $B_i = \{x_i^1, \ldots, x_i^{r_i}\}$.

The number of molecular operations in this program is of the order of $O(q \cdot p)$.

4.2 Formal Verification

In order to establish a formal verification of this molecular program we proceed to a re-labelling of the used tubes:

```
Procedure Cover
Input:  T_0 (where T_0 is a (p+q,q)-library)
     for i = 1 to q do
          (T_i^+, T_i^-) ← separate(T_{i-1}, i)
          T*_{i,0} ← T_i^+
          for j = 1 to r_i do
               T*_{i,j} ← set(T*_{i,j-1}, q + x_i^j)
          end for
          T_i ← combine(T*_{i,r_i}, T_i^-)
     end for
```

Obviously, this program is equivalent to the above one in the following sense: both have the same input tubes, T_0, and produce output tubes (T_0 and T_q, respectively) with the same content.

We want to note that the semantic of this program is very close to the semantic of the problem it solves. Hence, the problems that appear when trying to establish its formal verification are mainly due to the use of molecular operations that modify the inner structure of the molecules (i.e. `set` operation), making it harder the analysis of their track along the execution.

Next, we define a function, namely $STEP$, that captures the evolution of each molecule after the execution of a step of the main loop.

Definition 3. Let i be such that $1 \leq i \leq q$. Let $\rho \in T_{i-1}$. We define $STEP(\rho, i)$ as follows:

$$STEP(\rho, i) = \begin{cases} \rho\,, & \text{if } i \notin \rho \\ \rho \cup (q + B_i)\,, & \text{if } i \in \rho \end{cases}$$

Where $q + B_i = \{q + x_i^j : 1 \leq j \leq r_i\}$.

Note that if $\tau = STEP(\rho, i)$ then $\rho_q = \tau_q$ and $\rho \subseteq \tau$. Moreover, the function $STEP$ is a total function; that is, it is defined over all possible input data.

Lemma 1. $\forall\, i\ (1 \leq i \leq q \rightarrow \forall\, \rho \in T_{i-1}\ (STEP(\rho, i) \in T_i))$

Proof. Let i be such that $1 \leq i \leq q$, and let $\rho \in T_{i-1}$.

- If $i \notin \rho$, then $\rho \in -(T_{i-1}, i) = T_i^- \subseteq T_i$, and since $STEP(\rho, i) = \rho$, we deduce that $STEP(\rho, i) \in T_i$.
- If $i \in \rho$, we define recursively $\rho_{(j)}$, for each $0 \leq j \leq r_i$ as follows:
  - $\rho_{(0)} = \rho$.
  - $\rho_{(j+1)} = \rho_{(j)} \cup \{q + x_i^{j+1}\}$.

  Let us see that $\forall\, j\ (0 \leq j \leq r_i \rightarrow \rho_{(j)} \in T_{i,j}^*)$. By induction on j.
  - The case $j = 0$ is trivial because $\rho_{(0)} = \rho \in +(T_{i-1}, i) = T_i^+ = T_{i,0}^*$.
  - Let $j < r_i$ such that $\rho_{(j)} \in T_{i,j}^*$. From definition of $\rho_{(j+1)}$ we deduce that $\rho_{(j+1)} \in \mathtt{set}(T_{i,j}^*, q + x_i^{j+1}) = T_{i,j+1}^*$.

Since $\rho_{(r_i)} = STEP(\rho, i)$, we obtain that $STEP(\rho, i) \in T_{i,r_i}^* \subseteq T_i$. □

Definition 4. Let $\sigma \in T_0$. We define recursively σ^i, with $0 \leq i \leq q$, as follows:

$$\sigma^i = \begin{cases} \sigma\,, & \text{if } i = 0 \\ STEP(\sigma^{i-1}, i)\,, & \text{if } 1 \leq i \leq q \end{cases}$$

From Lemma 1 it follows that $\sigma^i \in T_i$, for each $0 \leq i \leq q$.

Lemma 2. *For each i, j such that $1 \leq i \leq q$ and $1 \leq j \leq r_i$ we have $\forall\, \tau \in T_{i,j}^*\ \exists\, \rho \in T_i^+\ (\tau = \rho \cup (q + B_i^j))$, where $B_i^j = \{x_i^1, \ldots, x_i^j\}$.*

Proof. For a given i, the proof can be deduced by induction on j following the semantic structure of the program. □

Corollary 1. *Let i be such that $1 \leq i \leq q$. Then for every molecule $\tau \in T_{i,r_i}^*$ there exists a molecule $\rho \in T_{i-1}$ verifying $i \in \rho$ and $STEP(\rho, i) = \tau$.*

Proof. Let i be such that $1 \leq i \leq q$. Let $\tau \in T_{i,r_i}^*$. From Lemma 2 (with $j = r_i$) we deduce that there exists $\rho \in T_i^+ = +(T_{i-1}, i)$ such that $\tau = \rho \cup (q + B_i^j)$. Then, $\rho \in T_{i-1}$ and $i \in \rho$. Hence, $STEP(\rho, i) = \tau$. □

4.3 Soundness of the Program

We have to prove that every molecule in the output tube, T_q, encodes a valid solution of the problem. That is, if $\tau \in T_q$, then τ_q encodes a subfamily, $\mathcal{F}_\tau$, of $\mathcal{F}$ and τ_p encodes a subset, A_τ, of A, such that $A_\tau = \bigcup \mathcal{F}_\tau$. In other words, we have to prove that every molecule, τ, in the output tube, T_q, is consistent.

To prove the soundness of the program we consider the following formula $\theta(i)$

$$\forall\, \tau \in T_i \quad [\forall\, k\, (1 \leq k \leq i \wedge k \in \tau \rightarrow (q + B_k) \subseteq \tau) \wedge \\ \wedge \forall\, s\, (1 \leq s \leq p \wedge (q+s) \in \tau \rightarrow \exists\, k \in \tau\, (1 \leq k \leq i \wedge s \in B_k))]$$

That is, the formula $\theta(i)$ (with $i > 0$) means that after the execution of the i-th step of main loop, all molecules, τ, verifying $A_\tau = \bigcup\{B_r \in \mathcal{F}_\tau : 1 \leq r \leq i\}$ are in the tube T_i. For the case $i = 0$ we suppose that $B_0 = \emptyset$.

Theorem 1. *The formula $\theta(i)$ is an invariant of the main loop. That is, $\forall\, i\, (0 \leq i \leq q \rightarrow \theta(i))$*

Proof. By induction on i. The case $i = 0$ is trivial. Let $i < q$ be such that the formula $\theta(i)$ is true. Let $\tau \in T_{i+1} = T^*_{i+1,r_{i+1}} \cup T^-_{i+1}$.

- If $\tau \in T^-_{i+1}$, then $\tau \in T_i \wedge i+1 \notin \tau$. In this case we have $\tau \in T_i$ and $i+1 \notin \tau$. As $\tau \in T_i$, from the induction hypothesis we obtain that

  (a) $\forall\, k\, (1 \leq k \leq i \wedge k \in \tau \rightarrow (q + B_k) \subseteq \tau)$

  (b) $\forall\, s\, (1 \leq s \leq p \wedge (q+s) \in \tau \rightarrow \exists\, k \in \tau\, (1 \leq k \leq i \wedge s \in B_k))]$

  Having in mind that $i+1 \notin \tau$ we have

  (a') $\forall\, k\, (1 \leq k \leq i+1 \wedge k \in \tau \rightarrow (q + B_k) \subseteq \tau)$

  From (b) we directly obtain

  (b') $\forall\, s\, (1 \leq s \leq p \wedge (q+s) \in \tau \rightarrow \exists\, k \in \tau\, (1 \leq k \leq i+1 \wedge s \in B_k))]$

  So, the formula $\theta(i+1)$ is true.

- If $\tau \in T^*_{i+1,r_{i+1}}$, from Corollary 1 we deduce that there exists $\rho \in T_i$ such that $\rho \in T_i$ verifying $i+1 \in \rho$ and $STEP(\rho, i+1) = \tau$. Then $\tau = \rho \cup (q + B_{i+1})$.

- Let k be such that $1 \leq k \leq i+1$, and $k \in \tau$.
  - If $1 \leq k \leq i$, then having in mind that $k \in \rho$, from induction hypothesis we obtain that $q + B_k \subseteq \rho$. So, $(q + B_k) \subseteq \tau$.
  - If $k = i+1$, from $\tau = \rho \cup (q + B_{i+1})$ we deduce that $(q + B_k) \subseteq \tau$.
- Let s be such that $1 \leq s \leq p$ and $(q+s) \in \tau$.
  - If $q + s \in \rho$, by induction hypothesis there exists $k \in \rho$ such that $1 \leq k \leq i$ and $s \in B_k$. But $1 \leq k \leq i \leq p \wedge k \in \rho \rightarrow k \in \tau$. Hence, there exists $k \in \tau$ such that $1 \leq k \leq i+1$ and $s \in B_k$.
  - If $q + s \notin \rho$, from $\tau = \rho \cup (q + B_{i+1})$ we obtain that $s \in B_{i+1}$.

Therefore, the formula $\theta(i+1)$ is true. □

Corollary 2. *(Soundness) Every molecule, τ, in the output tube, T_q, is a consistent molecule. That is, $A_\tau = \bigcup \mathcal{F}_\tau$.*

Proof. As formula $\theta(q)$ is true, for every molecule $\tau \in T_q$ we obtain that:

(1) $\forall\, k\ (1 \leq k \leq q \wedge k \in \tau \rightarrow (q + B_k) \subseteq \tau)$

(2) $\forall\, s\ (1 \leq s \leq p \wedge (q+s) \in \tau \rightarrow \exists\, k \in \tau\ (1 \leq k \leq q \wedge s \in B_k))$

Let us see that $\bigcup \mathcal{F}_\tau = A_\tau$. For that, let $s \in \bigcup \mathcal{F}_\tau$. Then $1 \leq s \leq p$ and $\exists\, k \in \tau\ (1 \leq k \leq q \wedge s \in B_k)$. From (1) it can be obtained $(q + B_k) \subseteq \tau$. Hence $(q+s) \in \tau$ and $s \in A_\tau$. Reciprocally, let $s \in A_\tau$. Then $1 \leq s \leq p \wedge (q+s) \in \tau$. From (2) we can deduce that there exists $k \in \tau$ such that $1 \leq k \leq q$ and $s \in B_k$. Hence $s \in \bigcup \mathcal{F}_\tau$. Therefore, $\bigcup \mathcal{F}_\tau = A_\tau$. □

4.4 Completeness of the Program

The completeness of the molecular program designed to solve this problem can be expressed as follows: for every subfamily, $\mathcal{F}'$, of $\mathcal{F}$ there exists a molecule in the output tube encoding the ordered pair $(\mathcal{F}', \bigcup \mathcal{F}')$. That is, to establish the completeness of the program we must show that every molecule, σ, in the input tube, T_0, is modified along the execution to give a consistent molecule, σ^q, in the output tube, T_q.

To prove completeness of the program we consider the following formula $\delta(i)$

$$\forall\, \sigma \in T_0 \quad [\forall\, k\ (1 \leq k \leq i\ \wedge\ k \in \sigma^i \rightarrow (q + B_k) \subseteq \sigma^i)\ \wedge \\ \wedge \forall\, s\ (1 \leq s \leq p\ \wedge\ (q+s) \in \sigma^i \rightarrow \exists\, k \in \sigma^i\ (1 \leq k \leq i\ \wedge\ s \in B_k))]$$

That is, the formula $\delta(i)$ (with $i > 0$) means that every molecule in the initial tube, T_0, encodes a subfamily $\mathcal{F}_i'$ of $\mathcal{F}_i = \{B_1, \ldots, B_i\}$ verifying the following: the union $\bigcup \mathcal{F}_i'$ of the sets belonging to $\mathcal{F}_i'$ is contained in the subset of A whose characteristic function is $(\sigma^i(q+1), \ldots, \sigma^i(q+p))$.

Theorem 2. *The formula $\delta(i)$ is an invariant of the main loop. That is, $\forall\, i\ (1 \leq i \leq q \rightarrow \delta(i))$*

Proof. Let i be such that $1 \leq i \leq q$. Let $\sigma \in T_0$. From Lemma 1 it can be deduced that $\sigma^i \in T_i$. Since $\theta(i)$ is true, we have

(1) $\forall\, k\ (1 \leq k \leq i\ \wedge\ k \in \sigma^i \rightarrow (q + B_k) \subseteq \sigma^i)$

(2) $\forall\, s\ (1 \leq s \leq p\ \wedge\ (q+s) \in \sigma^i \rightarrow \exists\, k \in \sigma^i\ (1 \leq k \leq i\ \wedge\ s \in B_k))$

Hence, the formula $\delta(i)$ is true. □

Corollary 3. *(Completeness) Every molecule, σ, in the input tube, is a consistent molecule, σ^q, in T_q, at the end of the execution.*

Proof. Let $\sigma \in T_0$. Since formula $\delta(q)$ is true, it verifies:
(1) $\forall\, k\ (1 \leq k \leq q\ \wedge\ k \in \sigma^q \rightarrow (q + B_k) \subseteq \sigma^q)$
(2) $\forall\, s\ (1 \leq s \leq p\ \wedge\ (q+s) \in \sigma^q \rightarrow \exists\, k \in \sigma^q\ (1 \leq k \leq q\ \wedge\ s \in B_k))$
As in corollary 2, and using the molecule σ^q instead of τ, we obtain that $\bigcup \mathcal{F}_{\sigma^q} = A_{\sigma^q}$. That is, the molecule σ^q is consistent. □

5 The Set Covering Problem

The *Set Covering* problem is the following: *Given a finite set $A = \{1, ..., p\}$ and a finite family $\mathcal{F} = \{B_1, ..., B_q\}$ of subsets of A, determine all subfamilies of $\mathcal{F}$ covering A.*

A molecular program in the sticker model solving the *Set Covering* problem is the following one:

```
Procedure Set_Covering
Input:  T_0 (where T_0 is a (p+q,q)-library)
      T_0 ← Cover(T_0)
      for s = 1 to p do
           T_0 ← +(T_0, q+s)
      end for
```

The number of molecular operations spent by the procedure `Set_Covering` is of the order of $O(q \cdot p)$.

In order to establish a formal verification of this molecular program we proceed to a re-labelling of the used tubes:

```
Procedure Cover_Selection
Input:  T_0 (where T_0 is a (p+q,q)-library)
      T_0 ← Cover(T_0)
      for s = 1 to p do
           T_s ← +(T_{s-1}, q+s)
      end for
```

Obviously, this program is equivalent to the above one in the following sense: both have the same input tubes, T_0, and produce output tubes (T_0 and T_p, respectively) with the same content.

5.1 Soundness of the Program

We have to prove that every molecule in the output tube encodes a cover of A. For that, we consider the following formula:

$$\theta(s) \equiv \forall\, \tau \in T_s\ ((q + A^s) \subseteq \tau)$$

where $A^s = \{1, \ldots, s\}$, and $q + A^s = \{q + j :\ 1 \leq j \leq s\}$.

That is, the formula $\theta(s)$ means that every molecule, τ, of T_s verifies that $A^s \subseteq A_\tau$ (where A_τ is the subset of A associated with τ, in a similar way than we made for the *Generating Cover Families* problem).

Theorem 3. *The formula $\theta(s)$ is an invariant of the main loop. That is,* $\forall\, s\ (1 \leq s \leq p \rightarrow \theta(s))$

Proof. By induction on s. Let $\tau \in T_1 = +(T_0, q+1)$. Then $(q+1) \in \tau$. So, $(q + A^1) \subseteq \tau$.

Let $s < p$ be such that $s \geq 1$ and the result is true for s. Let $\tau \in T_{s+1} = +(T_s, q+s+1)$. Then $\tau \in T_s$ and $(q+s+1) \in \tau$. As $\tau \in T_s$, from induction hypothesis we have $(q + A^s) \subseteq \tau$. Hence, $(q + A^{s+1}) \subseteq \tau$. □

Corollary 4. *(Soundness) Every molecule, τ, in the output tube, T_p, encodes a cover of A. That is, for every molecule $\tau \in T_p$ we have $(q + A) \subseteq \tau$.*

Proof. It suffices to note that the formula $\theta(p)$ is true and $A^p = A$. □

5.2 Completeness of the Program

Next, we have to prove that every molecule in the initial tube encoding a cover of A belongs to the output tube. For that, we consider the following formula:

$$\delta(s) \equiv \forall\, \sigma \in T_0\ ((q + A) \subseteq \sigma \rightarrow \sigma \in T_s)$$

That is, the formula $\delta(s)$ means that every molecule in T_0 encoding a cover of A belongs to the tube T_s.

Theorem 4. *The formula $\delta(s)$ is an invariant of the main loop. That is,* $\forall\, s\ (1 \leq s \leq p \rightarrow \delta(s))$

Proof. By induction on s. Let $\sigma \in T_0$ be such that $(q + A) \subseteq \sigma$. Then $\sigma \in T_0 \wedge (q+1) \in \sigma$. So, $\sigma \in +(T_0, q+1) = T_1$.

Let $s < p$ be such that $s \geq 1$ and the result is true for s. Let $\sigma \in T_0$ be such that $(q+A) \subseteq \sigma$. From induction hypothesis we have $\sigma \in T_s$. Since $1 \leq s+1 \leq p$ we obtain that $(q+s+1) \in \sigma$. Hence, $\sigma \in +(T_s, q+s+1)$. That is, $\sigma \in T_{s+1}$. □

Corollary 5. *Every molecule in the initial tube encoding a cover of A belongs to the output tube. That is, $\forall\, \sigma \in T_0\ ((q + A) \subseteq \sigma \rightarrow \sigma \in T_p)$.*

6 The Minimal Set Cover Selection Problem

The selection problem associated with the Minimal Set Cover consists, basically, in sorting, according to their cardinality, a family of covers of a given finite set. The introduction of *sticker model* by S. Roweis et al ([6]) is illustrated by presenting in this model a solution to the minimal set cover problem ([2]). Molecular solution given in [6] is founded in a subroutine that solves the above selection problem. We study a rooted graph structure that arise through the execution of the subroutine, and then we apply the above structured method to establish its formal verification.

The *Minimal Set Cover Selection* problem is the following: *Given a finite set* $A = \{1, ..., p\}$ *and a finite family* $\mathcal{F} = \{B_1, ..., B_q\}$ *of subsets of* A*, determine a subfamily of* $\mathcal{F}$ *covering* A*, and with minimal cardinality.*

For that, first we sort the collection of all subfamilies of $\mathcal{F}$ covering A, according to their cardinality and next we will select a minimal subfamily.

A molecular program in the sticker model solving the *Minimal Set Cover Selection* problem is the following one:

```
Input:  T_0 (encoding all sub-families of F
            covering A)
     For i ← 0 to q − 1 do
          T_{i+1} ← ∅
          For j ← i to 0 do
               T_j^+ ← +(T_j, i + 1) ; T_j^− ← −(T_j, i + 1)
               T_{j+1} ← combine(T_j^+, T_{j+1})
               T_j ← T_j^−
     if T_1 is nonempty Read T_1
     else read T_2
     else read T_3
     ⋮
     else read T_q
```

The number of molecular operations in this program is quadratic in the size of the family $\mathcal{F}$.

As in the previous problem, for each i such that $1 \leq i \leq q$ we will denote by r_i the cardinality of the set B_i, and $B_i = \{x_i^1, \ldots, x_i^{r_i}\}$.

Note that the input tube is not a library (as usual in the sticker model), since this program is in fact a subroutine. Moreover, this program does not enclose operations modifying inner structure of the strands (`set`, turn on, or `clear`, turn off). So, this subroutine can also be described in any molecular computation model without memory and based in filtering procedure.

Furthermore, if $\sigma \in T_0$, then $(\sigma(1), \ldots, \sigma(q))$ encodes in a natural way a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ as follows: $\forall i \ (1 \leq i \leq q \rightarrow (B_i \in \mathcal{F}' \leftrightarrow \sigma(i) = 1))$. Also, $\forall j \ (q+1 \leq j \leq q+p \rightarrow \sigma(j) = 1)$. We define $|\sigma| = \sum_{1 \leq i \leq q} \sigma(i)$.

According to this, the input tube used in this subroutine can be unloaded in the following way: we can use a restriction endonuclease enzyme so that when encoding the memory strands of initial test tube (that is, a $(p+q, q)$-library), an appropriate recognition site associated with the restriction enzyme is placed between q-th and $(q+1)$-th regions. In this way, just before running this subroutine, we activate the restriction endonuclease enzyme to make all memory strands split in the specific recognition site. After that, we select all molecules containing the first q regions (for example, using magnetic beads). Therefore we make that the input tube of this subroutine (noted here as T_0, too) contains all q regions length molecules encoding subfamilies of $\mathcal{F}$ covering A.

6.1 Formal Verification of the Subroutine

According to the exposed methodology, in order to establish the formal verification of the designed subroutine, we begin with a re-labelling procedure of tubes that have been used along the execution.

```
Input:  T_0
      T_{-1,0} ← T_0 ; T_{-1,1} ← ∅ ; T_{0,1} ← T_0 ; T*_{0,1} ← ∅
      For i ← 0 to q − 1 do
            T_{i,i+2} ← ∅
            For j ← i to 0 do
                  T+_{i,j} ← +(T_{i−1,j}, i + 1)
                  T_{i,j+1} ← combine(T+_{i,j}, T*_{i,j+1})
                  T*_{i,j} ← −(T_{i−1,j}, i + 1)
            T_{i,0} ← T*_{i,0}
      if T_{q−1,1} is nonempty Read T_{q−1,1}
      else Read T_{q−1,2}
      else Read T_{q−1,3}
      ⋮
      else Read T_{q−1,q}
```

Obviously, this program is equivalent to the above one in the following sense: both have the same input tubes, T_0, and produce output tubes ($T_0, \ldots, T_q$, and $T_{q-1,0}, \ldots, T_{q-1,q}$ respectively) such that the contents of the tubes T_i and $T_{q-1,i}$ (for $i = 0, \ldots, q$) are the same.

This program produces $i + 1$ tubes $(T_{i,i+1}, \ldots, T_{i,0})$ after the execution of the i-th step of the main loop, as Figure 1 illustrates.

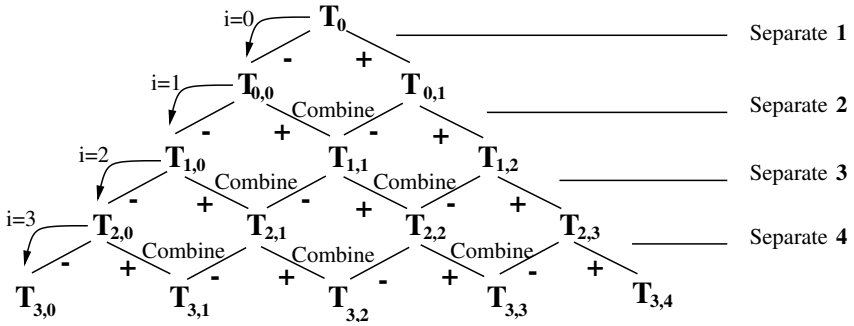

Figure 1. Labelled merge-binary tree.

That is, the execution of this program can be described as a rooted directed graph with depth q that we will call as *labelled merge-binary tree* with depth q, and that can be defined as follows:

Definition 5. A grid with depth h, $G_h = (V_h, E_h)$, is the following directed graph:

$$\begin{cases} V_h = \{(i,j) :\ 0 \leq i \leq h \wedge 0 \leq j \leq i\} \\ E_h = \{((i,j),(i+1,j)),\ ((i,j),(i+1,j+1)) :\ 0 \leq i < h \wedge 0 \leq j \leq i\} \end{cases}$$

Definition 6. A labelled merge-binary tree with depth h is a tuple

$$(G_h, L, \{F_i :\ 0 \leq i \leq h-1\}, B, T_0, l)$$

where:

- G_h is a grid with depth h.
- L is a nonempty set (its elements will be called *labels*).
- For each i (with $0 \leq i \leq h-1$) F_i is a function from L to $L \times L$ (we will note $F_i = (F_i^-, F_i^+)$).
- B is a binary function from $L \times L$ to L.
- T_0 belongs to L (that is, T_0 is a label).
- l is a function from V_h to L, called the *labelling function*, defined by recursion (using T_0, F_i and B), as follows:

$$\begin{cases} l(0,0) = T_0 \\ l(i+1,0) = F_i^-(l(i,0)) \\ l(i+1,i+1) = F_i^+(l(i,i)) \\ l(i+1,j) = B(F_i^-(l(i,j-1)), F_i^+(l(i,j))) \end{cases}$$

  with $0 \leq i < h$ and $1 \leq j \leq i$.

In the execution of the designed molecular program a labelled merge-binary tree with depth q is obtained, where:

(a) The labels are the tubes.

(b) For each i (with $0 \leq i \leq q-1$) we have $F_i(T) = \texttt{separate}(T, i+1)$. So,

$$F_i^-(T) = -(T, i+1) \text{ and } F_i^+(T) = +(T, i+1)$$

(c) The binary function, B, is the `combine` molecular operation (called `merge` too).

(d) T_0 is the input test–tube of the subroutine.

This combinatorics structure allows us to design a subroutine solving a more general sorting problem, where the semantic of the subroutine is very close to the semantic of the problem ([3]).

To establish the formal verification of the subroutine, in connection with the *Minimal Set Cover Selection* problem, we have to prove specifically that:

- Every molecule, σ, in the output tube, $T_{q-1,r}$ (with $1 \leq r \leq q$), verifies $|\sigma| = r$ (*Soundness*).
- Every molecule, σ, in the input tube, T_0, such that $|\sigma| = r$ (with $1 \leq r \leq q$), is in the output tube $T_{q-1,r}$ (*Completeness*).

The execution of the subroutine can be seen as an evolution of a population of elements. Initially, the population is determined by the multiset of molecules in the input tube, T_0. Every molecule is an individual, and repeated ones can exist at the same time (so cloned members can be alive simultaneously in this population).

Every step of the main loop can be interpreted as a time unit. After a lapse, the population is transformed into another one, but, in this case, there is no death or mutation in elements, since this subroutine is, basically, a filtering procedure.

We consider the following formulas:

$$\begin{aligned} \psi(i,0) &\equiv \forall\sigma\,(\sigma \in T_{i,0} \leftrightarrow \sigma \in T_0 \wedge \forall k\,(1 \leq k \leq i+1 \rightarrow k \notin \sigma)) \\ \psi(i,j+1) &\equiv \forall\sigma(\sigma \in T_{i,j+1} \leftrightarrow (\sigma \in T_{i-1,j} \wedge i+1 \in \sigma) \vee \\ &\qquad \vee\,(\sigma \in T_{i-1,j+1} \wedge i+1 \notin \sigma)) \\ \theta(i) &\equiv \forall j\,(0 \leq j \leq i+1 \rightarrow \psi(i,j)) \end{aligned}$$

Theorem 5. *The formula $\theta(i)$ is an invariant of the main loop. That is, $\forall\, i\,(0 \leq i \leq q-1 \rightarrow \theta(i))$.*

Proof. By induction on i.

- The result is true for $i = 0$.

  The formula $\psi(0,0)$ is true because

$\sigma \in T_{0,0} \Longleftrightarrow \sigma \in T^*_{0,0} = -(T_{-1,0}, 1) = -(T_0, 1) \Longleftrightarrow$
$\sigma \in T_0 \wedge 1 \notin \sigma \Longleftrightarrow \sigma \in T_0 \wedge \forall k\, (1 \leq k \leq 0+1 \rightarrow k \notin \sigma)$.

The formula $\psi(0,1)$ is true because

$\sigma \in T_{0,1} \Longleftrightarrow \sigma \in T^+_{0,0} \cup T^*_{0,1} = T^+_{0,0} = +(T_{-1,0}, 1) \Longleftrightarrow \sigma \in$
$T_{-1,0} \wedge 1 \in \sigma \Longleftrightarrow (\sigma \in T_{-1,0} \wedge 1 \in \sigma) \vee (\sigma \in T_{-1,1} \wedge 1 \notin \sigma)$.

- Let $i < q-1$ such that the formula $\theta(i)$ is true. Let us see that the formula $\theta(i+1)$ is true; that is, $\forall\, j\ (0 \leq j \leq i+2 \rightarrow \psi(i+1, j)$.
  - For $j = 0$ we have the following:
    $\sigma \in T_{i+1,0} \Longleftrightarrow \sigma \in T^*_{i+1,0} \Longleftrightarrow \sigma \in -(T_{i,0}, i+2) \Longleftrightarrow \sigma \in$
    $T_{i,0} \wedge i+2 \notin \sigma \Longleftrightarrow \sigma \in T_0 \wedge \forall k\, (1 \leq k \leq i+1 \rightarrow k \notin$
    $\sigma) \wedge i+2 \notin \sigma \Longleftrightarrow \sigma \in T_0 \wedge \forall k\, (1 \leq k \leq i+2 \rightarrow k \notin \sigma)$.
  - Let $j \leq i+2$ such that $j > 0$ and let us see that $\psi(i+1, j)$ is true. Indeed:
    $\sigma \in T_{i+1,j} \Longleftrightarrow \sigma \in T^+_{i+1,j-1} \cup T^*_{i+1,j} \Longleftrightarrow \sigma \in +(T_{i,j-1}) \cup$
    $-(T_{i,j}, i+2) \Longleftrightarrow (\sigma \in T_{i,j-1} \wedge i+2 \in \sigma) \vee (\sigma \in T_{i,j} \wedge i+2 \notin \sigma)$.

□

Next, we describe the trace of every molecule in the input tube along the execution of the subroutine. For this, if $\sigma \in T_0$ is given, we will write $\sigma = (i_1, \ldots, i_r)$ to note that $1 \leq i_1 < \cdots < i_r \leq q$ and

$$\forall\, j\ (1 \leq j \leq r \rightarrow \sigma(i_j) = 1) \wedge \forall\, t\, \forall\, j\ (1 \leq t \leq q \wedge i_j \neq t \rightarrow \sigma(t) = 0)$$

That is, the molecule $\sigma = (i_1, \ldots, i_r) \in T_0$ encodes in a natural way the subfamily $\mathcal{F}' = \{B_{i_1}, \ldots, B_{i_r}\}$ of $\mathcal{F}$.

Proposition 1. *Let $\sigma = (i_1, \ldots, i_r) \in T_0$, where $1 \leq i_1 < i_2 < \cdots < i_r \leq q$. Then*

(1) $\forall j\ (1 \leq j \leq r \rightarrow \sigma \in T_{i_j - 1, j})$.

(2) $\forall t\ (1 \leq t \leq q - i_r \rightarrow \sigma \in T_{i_r + t - 1, r})$.

(3) $\sigma \in T_{q-1,r}$.

Proof.

1. By induction on j.

   - First, we consider the case $j = 1$.
     - If $i_1 = 1$ then $T_{i_1-1,1} = T_{0,1} = T_0$. So, $\sigma \in T_{i_1-1,1}$.
     - If $i_1 > 1$ then we prove that $\forall\, s\ (1 \leq s < i_1 \rightarrow \sigma \in T_{s-1,0})$. Indeed:
       * Let s be such that $1 \leq s < i_1$. Then, $\sigma \in T_0 \wedge \forall\, k\ (1 \leq k \leq s - 1 + 1 \rightarrow k \notin \sigma)$. As $s - 1 \geq 0$ the formula $\psi(s-1, 0)$ is true, so we deduce that $\sigma \in T_{s-1,0}$.

       From $1 \leq i_1 - 1 < i_1$ we obtain that $\sigma \in T_{(i_1-1)-1,0}$. As $i_1 \in \sigma$ we have $\sigma \in +(T_{i_1-2,0}, i_1)$. Then $\sigma \in T_{i_1-2,0}$, $i_1 \in \sigma$, and the formula $\psi(i_1-1, 1)$ is true. Hence, $\sigma \in T_{i_1-1,1}$.
   - Let j be such that $1 \leq j < r$. Let us suppose that $\sigma \in T_{i_j-1,j}$. We have to prove that $\sigma \in T_{i_{j+1}-1,j+1}$.
     - If $i_{j+1} - 1 = i_j$ (that is, $i_{j+1} = i_j + 1$), from induction hypothesis we have $\sigma \in T_{i_j-1,j}$ and $i_{j+1} \in \sigma$. That is, $\sigma \in T_{i_j-1,j}$ and $i_j+1 \in \sigma$. As formula $\psi(i_{j+1}-1, j+1) \equiv \psi(i_j, j+1)$ is true, we deduce that $\sigma \in T_{i_{j+1}-1,j+1}$.
     - If $i_{j+1} - 1 > i_j$ we prove that $\forall\, t\ (1 \leq t \leq i_{j+1} - i_j - 1 \rightarrow \sigma \in T_{i_j+t-1,j})$, by induction on t.
       * We have $\sigma \in T_{i_j-1,j}$ and $i_j + 1 \notin \sigma$. As formula $\psi(i_j, j)$ is true (and $j > 0$) we obtain that $\sigma \in T_{i_j,j}$. That is, $\sigma \in T_{i_j+1-1,j}$. So, the result is true for $t = 1$.
       * Let t be such that $t < i_{j+1} - i_j - 1$ and let us suppose the result holds for t. Then, $\sigma \in T_{i_j+t-1,j}$. As $i_j < i_j + t + 1 < i_{j+1}$, we have $i_j + t + 1 \notin \sigma$. Having in mind that the formula $\psi(i_j+t, j)$ is true, we conclude that $\sigma \in T_{i_j+t,j}$. That is, $\sigma \in T_{i_j+(t+1)-1,j}$.

     Now, let us see that $\sigma \in T_{i_{j+1}-1,j+1}$. Applying the above relation for $t = i_{j+1} - i_j - 1$, $\sigma \in T_{i_j+(i_{j+1}-i_j-1)-1,j} = T_{i_{j+1}-2,j}$. As $i_{j+1} \in \sigma$ and the formula $\psi(i_{j+1} - 1, j + 1)$ is true we deduce that $\sigma \in T_{i_{j+1}-1,j+1}$.

2. By induction on t.

- For $t = 1$ we have that $1 \leq q - i_r$, that is, $i_r + 1 \leq q$. From (1) we obtain that $\sigma \in T_{i_r-1,r}$. But $i_r + 1 \notin \sigma$. As formula $\psi(i_r, r)$ is true we conclude that $\sigma \in T_{i_r,r}$. That is, $\sigma \in T_{i_r+1-1,r}$.
- Let t be such that $1 \leq t < q - i_r$, and let us suppose that $\sigma \in T_{i_r+t-1,r}$. As $i_r < i_r + t + 1 \leq q$ we have $i_r + t + 1 \notin \sigma$. Taking into account that the formula $\psi(i_r + t, r)$ is true (because $i_r + t \leq q - 1$), we conclude that $\sigma \in T_{i_r+t,r}$. That is, $\sigma \in T_{i_r+(t+1)-1,r}$.

3. Applying the result obtained in (2), considering $t = q - i_r$, we deduce that $\sigma \in T_{i_r+(q-i_r)-1,r} = T_{q-1,r}$.

□

Next we will prove that the generated tubes after i-th step of the main loop, $\{T_{i,0}, T_{i,1}, \ldots, T_{i,i+1}\}$, form a *partition* of the initial test tube, T_0. This confirms that the subroutine runs a filtering procedure where no strand dies along the execution.

Proposition 2. $\forall i \ (0 \leq i \leq q - 1 \rightarrow T_0 = \bigcup_{0 \leq j \leq i+1} T_{i,j})$.

Proof. Let us first prove that $\forall\, i \ (0 \leq i \leq q - 1 \rightarrow T_0 \subseteq \bigcup_{0 \leq j \leq i+1} T_{i,j})$. By induction on i.

- Let $\sigma \in T_0$. Then $(\sigma \in T_0 \wedge 1 \in \sigma) \vee (\sigma \in T_0 \wedge 1 \notin \sigma)$. So,

  $$(\sigma \in T_0 \wedge 1 \in \sigma) \Longrightarrow (\sigma \in T_{-1,0} \wedge 1 \in \sigma) \Longrightarrow \sigma \in T_{0,1}$$
  $$(\sigma \in T_0 \wedge 1 \notin \sigma) \Longrightarrow \sigma \in T_{0,0}$$

  Hence $\sigma \in T_{0,1} \cup T_{0,0} \subseteq \bigcup_{0 \leq j \leq 1} T_{0,j}$.

- Let $i < q - 1$ be such that $T_0 \subseteq \bigcup_{0 \leq j \leq i+1} T_{i,j}$. Let $\sigma \in T_0$. From induction hypothesis, there exists j such that $0 \leq j \leq i + 1$ and $\sigma \in T_{i,j}$. Then $(\sigma \in T_{i,j} \wedge i + 2 \in \sigma) \vee (\sigma \in T_{i,j} \wedge i + 2 \notin \sigma)$.

  – Let us suppose that $\sigma \in T_{i,j}$ and $i + 2 \in \sigma$.
  As formula $\psi(i+1, j+1)$ is true we deduce that $\sigma \in T_{i+1,j+1}$ $\subseteq \bigcup_{0 \leq s \leq i+2} T_{i+1,s}$.

  – Let us suppose that $\sigma \in T_{i,j}$ and $i + 2 \notin \sigma$

- If $j = 0$ then $\sigma \in T_{i,0}$ and $i+2 \notin \sigma$. But $\sigma \in T_{i,0} \Longrightarrow \sigma \in T_0 \wedge \forall\, k\ (1 \leq k \leq i+1 \rightarrow k \notin \sigma)$. So, $\sigma \in T_0 \wedge \forall\, k\ (1 \leq k \leq i+2 \rightarrow k \notin \sigma)$. As formula $\psi(i+1,0)$ is true, we deduce that $\sigma \in T_{i+1,0} \subseteq \bigcup_{0 \leq s \leq i+2} T_{i+1,s}$.
- If $j > 0$, taking into account that $\sigma \in T_{i,j} \wedge i+2 \notin \sigma$ and the formula $\psi(i+1,j)$ is true, we conclude that $\sigma \in T_{i+1,j}$. That is, $\sigma \in \bigcup_{0 \leq s \leq i+2} T_{i+1,s}$.

Now, let us see that $\forall\, i\ (0 \leq i \leq q-1 \rightarrow \bigcup_{0 \leq j \leq i+1} T_{i,j} \subseteq T_0)$. For that, it is enough to prove that $\forall\, i \forall\, j\ (0 \leq i \leq q-1 \wedge 0 \leq j \leq i+1 \rightarrow T_{i,j} \subseteq T_0)$. By induction on i.

- The result holds for $i = 0$. Indeed:

$$T_{0,0} = T^*_{0,0} = -(T_{-1,0}, 1) = -(T_0, 1) \subseteq T_0$$
$$T_{0,1} = T^+_{0,0} \cup T^*_{0,1} = T^+_{0,0} = +(T_{-1,0}, 1) = +(T_0, 1) \subseteq T_0$$

- Let $i < q-1$ be such that $\forall\, j\ (0 \leq j \leq i+1 \rightarrow T_{i,j} \subseteq T_0)$. Let us see that $\forall\, j\ (0 \leq j \leq i+2 \rightarrow T_{i+1,j} \subseteq T_0)$.
  - If $j = 0$ then $T_{i+1,0} = T^*_{i+1,0} = -(T_{i,0}, i+2) = -(T_{i,0}, i+2) \overset{h.i.}{\subseteq} T_0$.
  - If $j > 0$ (and $j \leq i+2$) then we have: $T_{i+1,j} = T^*_{i+1,j-1} \cup T^*_{i+1,j} = +(T_{i,j-1}, i+2) \cup -(T_{i,j}, i+2) \subseteq T_{i,j-1} \cup T_{i,j}$. Taking into account that

$$j \leq i+1 \Longrightarrow T_{i,j-1} \cup T_{i,j} \subseteq T_0$$
$$j = i+2 \Longrightarrow T_{i,j-1} \cup T_{i,j} = T_{i,i+1} \cup T_{i,i+2} = T_{i,i+1} \subseteq T_0$$

    we conclude that $T_{i+1,j} \subseteq T_0$.

□

Proposition 3. $\forall i\ (0 \leq i \leq q-1 \rightarrow \forall r\ \forall s\ (0 \leq r < s \leq i+1 \rightarrow T_{i,r} \cap T_{i,s} = \emptyset))$.

Proof. By induction on i.

- The result holds for $i = 0$ because

$$T_{0,0} \cap T_{0,1} = T^*_{0,0} \cap T_{0,1} = -(T_{-1,0}, 1) \cap +(T_{-1,0}, 1) = \emptyset.$$

- Let $i < q-1$ be such that $\forall\, r\ \forall\, s\ (0 \leq r < s \leq i+1 \rightarrow T_{i,r} \cap T_{i,s} = \emptyset)$. Let r, s be such that $0 \leq r < s \leq i+2$. Let us see that $T_{i+1,r} \cap T_{i+1,s} = \emptyset$.

  – Let us suppose that $s = i+2$.
  If there exists $\tau \in T_{i+1,r} \cap T_{i+1,i+2}$ then

  $$\begin{array}{rcl} \tau \in T_{i+1,i+2} & \Longrightarrow & (\tau \in T_{i,i+1} \wedge i+2 \in \tau) \vee \\ & & (\tau \in T_{i,i+2} \wedge i+2 \notin \tau) \\ & \Longrightarrow & (\tau \in T_{i,i+1} \wedge i+2 \in \tau) \end{array}$$

  But $\tau \in T_{i+1,0} \Longrightarrow \tau \in T_0 \wedge \forall\, k\ (1 \leq k \leq i+2 \rightarrow k \notin \tau)$, and $i+2 \in \tau$. So, $r > 0$. Then, $\tau \in T_{i+1,r} \Longrightarrow \tau \in T_{i,r-1}$. Hence, $T_{i,i+1} \cap T_{i,r-1} \neq \emptyset$. This contradicts our assumption.

  – Let us suppose that $r = 0$ and $1 \leq s \leq i+1$.
  If there exists $\tau \in T_{i+1,0} \cap T_{i+1,s}$ then

  $$\begin{array}{rcl} \tau \in T_{i+1,0} & \Longrightarrow & \tau \in T_0 \wedge \forall\, k\ (1 \leq k \leq i+2 \rightarrow k \notin \tau) \\ & \Longrightarrow & \tau \in T_{i,0} \wedge \forall\, k\ (1 \leq k \leq i+2 \rightarrow k \notin \tau) \\ \tau \in T_{i+1,s} & \Longrightarrow & (\tau \in T_{i,s-1} \wedge \leq i+2 \in \tau) \vee \\ & & (\tau \in T_{i,s} \wedge i+2 \notin \tau) \end{array}$$

  So, $\tau \in T_{i,0} \cap T_{i,s}$. This contradicts the induction hypotesis (because $0 < s \leq i+1$).

  – Let us suppose that $r > 0$ and $1 \leq s \leq i+1$.
  If there exists $\tau \in T_{i+1,r} \cap T_{i+1,s}$ then

  $$\begin{array}{rcl} \tau \in T_{i+1,r} & \Longrightarrow & (\tau \in T_{i,r-1} \wedge i+2 \in \tau) \vee \\ & & (\tau \in T_{i,r} \wedge i+2 \notin \tau) \\ \tau \in T_{i+1,s} & \Longrightarrow & (\tau \in T_{i,s-1} \wedge i+2 \in \tau) \vee \\ & & (\tau \in T_{i,s} \wedge i+2 \notin \tau) \end{array}$$

  If $i+2 \in \tau$ then $T_{i,r-1} \cap T_{i,s-1} \neq \emptyset$, which contradicts the induction hypothesis.
  If $i+2 \notin \tau$ then $T_{i,r} \cap T_{i,s} \neq \emptyset$, which contradicts the induction hypothesis.

□

Corollary 6. $T_{q-1,0} = \emptyset$.

Proof. Let us suppose that $T_{q-1,0} \neq \emptyset$. Then there exists $\sigma = (i_1, \ldots, i_r) \in T_{q-1,0}$, with $r > 0$. From Proposition 1.(3) we have $\sigma \in T_{q-1,r}$. So, $T_{q-1,0} \cap T_{q-1,r} \neq \emptyset$, which contradicts Proposition 3. □

Finally, we establish the soundness and the completeness of the designed subroutine that solves the *Minimal Set Cover Selection* problem.

Theorem 6. `(Soundness)` *Every strand, σ, in the final test–tube, $T_{q-1,r}$ (with $1 \leq r \leq q$), verifies that its length is r. That is, $\forall r\ (1 \leq r \leq q \rightarrow \forall \sigma \in T_{q-1,r}(|\sigma| = r))$.*

Proof. Let r be such that $1 \leq r \leq q$. Let $\sigma \in T_{q-1,r}$. From Proposition 2 we obtain that $\sigma \in T_0$. From Proposition 1.(3), we have $\sigma \in T_{q-1,|\sigma|}$. From Proposition 3 we conclude that $|\sigma| = r$. □

Theorem 7. `(Completeness)` *Every strand, σ, in the initial test–tube, T_0, with length r is in the output tube, $T_{q-1,r}$. That is, $\forall\, \sigma \in T_0\ (|\sigma| = r \rightarrow \sigma \in T_{q-1,r})$. Moreover, $\forall \sigma \in T_0\ \exists! r\ (1 \leq r \leq q \wedge \sigma \in T_{q-1,r})$.*

Proof. Let $\sigma \in T_0$ such that $|\sigma| = r$. From Proposition 1.(3) we can deduce that $\sigma \in T_{q-1,r}$. In the other hand, from Proposition 3 (with $i = q - 1$) there exists j such that $0 \leq j \leq i + 1$ and $\sigma \in T_{q-1,j}$. From Corollary 1 we obtain $j > 0$, and from Proposition 3 we conclude that j is unique. □

7 Conclusions

A methodology to establish the formal verification of FOR or $WHILE$ programs within molecular computing models with random access memory has been presented. The search of invariant formulas of the main loop needs a detailed study of the evolution of every molecule along the execution of the program. A function called $STEP$ has been introduced to capture the evolution of the molecules after one step of the main loop.

These solutions are applied to solve some numerical **NP**-complete problems: the *Generating Cover Families* problem, the *Set Covering* problem and the *Minimal Set Cover* Selection Problem. The problems we

have chosen in this paper present some interesting and different characteristics with respect to the $STEP$ function.

We think that the study of the formal verification of molecular programs is a necessary step for their automatic processing by means of reasoning systems (we are currently working on ACL2 and PVS).

Acknowledgement

The support for this research through the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds, is gratefully acknowledged.

References

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

[2] Garey, M. R.; Johnson, D. S. *Computers and intractability*, W. H. Freeman and Company, New York, 1979.

[3] Pérez-Jiménez, M. J.; Sancho-Caparrini, F. Solving Knapsack Problems in a Sticker Based Model, in Jonoska, N.; Seeman, N. (eds.), *DNA Computing*, LNCS **2340** (2002), 161–171.

[4] Hoare, C. A. R. An axiomatic basis for computer programming, *Communications of the ACM*, **12** (1969), 576–583.

[5] Pérez-Jiménez, M. J.; Sancho-Caparrini, F. Minimal Set Cover Problem: On a DNA solution of selection stage, in Martín-Vide, C.; Păun, Gh. (eds.), *Pre-Proceedings of Workshop on Membrane Computing*, RGML Report **17/01** (2001), 251–258.

[6] Roweis, S.; Winfree, E.; Burgoyne, R.; Chelyapov, N.; Goodman, M.; Rothemund, P.; Adleman, L. A Sticker–Based Model for DNA Computation, *Journal of Computational Biology*, **5**, 4 (1998), 615–629.