STUDIA UNIV. BABEŞ-BOLYAI, INFORMATICA, VOLUME XLIV, NUMBER 1, 1999

THEOREM PROVING AND DNA COMPUTING

DOINA TATAR AND MIHAI OLTEAN

Abstract. We start from sticker systems as language generating devices for DNA computing ([3]) and we define the sticker systems associate with a set of clauses. The Robinson's theorem is stated in terms of the language generated by this sticker system.

1. Introduction

A DNA (dezoxyribose nucleic acid) is a large double-stranded (helicoidal) structure that contains, in order form, all information needed to generate proteins for living organisms. This coded information is a sequence of four nucleotides, A (adenine), T (Thymine), C (Cytosine), G (Guanine) paired A-T, C-G according to the so-called Watson-Crick complementarity [3].

One can think DNA as a program interpreted by a complex biological machinery that generates sequence of aminoacids (proteins). There are precisely 20 aminoacids (proteins) that can be generated from 64 possible triplets (codons) of nucleotides, and each of them can be represented by multiple triplets[2]. For example, the aminoacid Ala can be formed by the following triplets: GCA, GCC, GCT,GCC. In a simplified manner, the generation of proteins form DNA proceeds in four phases: transcription, splicing, aminoacid generation and protein folding.

In [3], [4] the authors present a language-theoretic model of DNA splicing. This model will be adopted for resolution method in automated theorem proving. In the following section we will present the sticker operation and the sticker systems as introduced in [3].

In section 3 we will present the resolution method as a complete computation in a stiker system.

In section 4 the corresponding justifications in propositional calculus are introduced.

2. Sticker operation and sticker system

If we have single stranded sequence of A, C, G, T nucleotides, together with a single stranded sequence composed of the complementary nucleotides, the two sequences will be glued together (by hydrogens bounds), forming a double stranded DNA sequence. What [3] extracts from here is the operation of prolonging to the right a sequnce of (single or double) symbols by using given single stranded strings, matching them with portions of the current sequence accordind to a complementary relation.

In the following we will present the sticker operation and the sticker system as in [3],[4].

Let V be an alphabet endowed with a symmetric relation ρ (of complementarity), $\rho \subseteq V x V$. Let # be a special symbol not in V, denoting an empty space (the blank symbol).

Using the elements of $V \cup \{\#\}$ one construct the composite symbols of the following sets:

$$\begin{pmatrix} V \\ V \end{pmatrix}_\rho = \left\{ \begin{pmatrix} a \\ b \end{pmatrix} \mid a, b \in V, (a,b) \in \rho \right\},$$

$$\begin{pmatrix} \# \\ V \end{pmatrix}_\rho = \left\{ \begin{pmatrix} \# \\ a \end{pmatrix} \mid a \in V \right\},$$

$$\begin{pmatrix} V \\ \# \end{pmatrix}_\rho = \left\{ \begin{pmatrix} a \\ \# \end{pmatrix} \mid a \in V \right\}.$$

Now the set:

$$W_\rho(V) = \begin{pmatrix} V \\ V \end{pmatrix}_\rho^* S(V),$$

where

$$S(V) = \begin{pmatrix} \# \\ V \end{pmatrix}^* \cup \begin{pmatrix} V \\ \# \end{pmatrix}^*,$$

is introduced and the elements of this set are called *well-started sequences*. S

Stated otherwise, the elements of W_ρ (V) start with pairs of symbols in V, as selected by the complementarity relation, and end:

a) either by a suffix consisting of pairs $\begin{pmatrix} \# \\ a \end{pmatrix}$

b) or with a suffix consisting of pairs $\begin{pmatrix} b \\ \# \end{pmatrix}$, for $a,b \in V$ (the symbols $\begin{pmatrix} \# \\ a \end{pmatrix}, \begin{pmatrix} b \\ \# \end{pmatrix}$ are not mixed).

The sticker operation, denoted by μ, is a partially defined mapping from $W_\rho(V)$ x $S(V)$ to $W_\rho(V)$ definde as follows: for $x \in W_\rho(V)$, $y \in S(V)$, $z \in W_\rho(V)$, one write

$\mu(x,y) = z$

if and only if one of the following cases holds:

1.

$$x = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} a_{k+1} \\ \# \end{pmatrix} \cdots \begin{pmatrix} a_{k+r} \\ \# \end{pmatrix} \begin{pmatrix} a_{k+r+1} \\ \# \end{pmatrix} \cdots \begin{pmatrix} a_{k+r+p} \\ \# \end{pmatrix},$$

$$y = \begin{pmatrix} \# \\ c_1 \end{pmatrix} \cdots \begin{pmatrix} \# \\ c_r \end{pmatrix}$$

$$z = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} a_{k+1} \\ c_1 \end{pmatrix} \cdots \begin{pmatrix} a_{k+r} \\ c_r \end{pmatrix} \begin{pmatrix} a_{k+r+1} \\ \# \end{pmatrix} \cdots \begin{pmatrix} a_{k+r+p} \\ \# \end{pmatrix}$$

for $k \geq 0$, $r \geq 1$, $p \geq 1$,
$a_i \in V$, $1 \leq i \leq k+r+p$, $b_i \in V$, $1 \leq i \leq k$, $c_i \in V$, $1 \leq i \leq r$,
and $(a_{k+i}, c_i) \in \rho$, $1 \leq i \leq r$;

2.

$$x = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} a_{k+1} \\ \# \end{pmatrix} \cdots \begin{pmatrix} a_{k+r} \\ \# \end{pmatrix}$$

$$y = \begin{pmatrix} \# \\ c_1 \end{pmatrix} \cdots \begin{pmatrix} \# \\ c_r \end{pmatrix} \begin{pmatrix} \# \\ c_{r+1} \end{pmatrix} \cdots \begin{pmatrix} \# \\ c_{r+p} \end{pmatrix},$$

$$z = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} a_{k+1} \\ c_1 \end{pmatrix} \cdots \begin{pmatrix} a_{k+r} \\ c_r \end{pmatrix} \begin{pmatrix} \# \\ c_{r+1} \end{pmatrix} \cdots \begin{pmatrix} \# \\ c_{r+p} \end{pmatrix},$$

for $k \geq 0$, $r \geq 1$, $p \geq 1$,
$a_i \in V$, $1 \leq i \leq k+r+p$, $b_i \in V$, $1 \leq i \leq k$, $c_i \in V$, $1 \leq i \leq r$,
and $(a_{k+i}, c_i) \in \rho$, $1 \leq i \leq r$;

3.

$$x = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} \# \\ b_{k+1} \end{pmatrix} \cdots \begin{pmatrix} \# \\ b_{k+r} \end{pmatrix} \begin{pmatrix} \# \\ b_{k+r+1} \end{pmatrix} \cdots \begin{pmatrix} \# \\ b_{k+r+p} \end{pmatrix},$$

$$y = \begin{pmatrix} c_1 \\ \# \end{pmatrix} \cdots \begin{pmatrix} c_r \\ \# \end{pmatrix},$$

$$z = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} \cdots \begin{pmatrix} a_k \\ b_k \end{pmatrix} \begin{pmatrix} c_1 \\ b_{k+1} \end{pmatrix} \cdots \begin{pmatrix} c_k \\ b_{k+r} \end{pmatrix} \begin{pmatrix} \# \\ b_{k+r+1} \end{pmatrix} \cdots \begin{pmatrix} \# \\ b_{k+r+p} \end{pmatrix},$$

for $k \geq 0$, $r \geq 1$, $p \geq 1$,
$a_i \in V$, $1 \leq i \leq k$, $b_i \in V$, $1 \leq i \leq k+r+p$, $c_i \in V$, $1 \leq i \leq r$,
and $(a_{k+i}, c_i) \in \rho$, $1 \leq i \leq r$;

4.

$$y = \binom{c_1}{\#}\cdots\binom{c_r}{\#}\binom{c_{r+1}}{\#}\cdots\binom{c_{r+p}}{\#},$$

$$x = \binom{a_1}{b_1}\cdots\binom{a_k}{b_k}\binom{\#}{b_{k+1}}\cdots\binom{\#}{b_{k+r}}$$

$$z = \binom{a_1}{b_1}\cdots\binom{a_k}{b_k}\binom{c_1}{b_{k+1}}\cdots\binom{c_k}{b_{k+r}}\binom{c_{r+1}}{\#}\cdots\binom{c_{r+p}}{\#},$$

for $k \geq 0,\ r \geq 0,\ r+p \geq 1,$

$a_i \in V,\ 1 \leq i \leq k,\ b_i \in V,\ 1 \leq i \leq k+r,\ c_i \in V,\ 1 \leq i \leq r+p,$

and $(a_{k+i}, c_i) \in \rho,\ 1 \leq i \leq r;$

In the case 1 one add complementary symbols on the lower level without completing all the blank spaces. In case one we complete the blank spaces on the lower level of x and possibly add more composite symbols of the form $\binom{\#}{c}$. Cases 3 and 4 are symmetric to cases 1 and 2, respectively, completing blank spaces on the upper level of the string.

Note that in all cases the string y must contain at least one composite symbol and that cases 2 and 4 allow the prolongation on of "blunt" strings in $W_\rho(V)$: when r = 0 there is no blank position in x.

For strings x, y which do not satisfy any of the previous conditions, $\mu(x,y)$ is not defined.

Using the sticker operation in [3] the authors define a generating/computing mechanism, sticker system, as follows:

$\gamma = (V, \rho, A, B_d, B_u)$,

where V is an alphabet, $\rho \subseteq VxV$ is a symetric relation on V, A is a finite subset of $W_\rho(V)$ of axioms, and B_d,

B_u are finite subsets of $\binom{V}{\#}^+$ and $\binom{\#}{V}^+$, respectively.

The idea behind such a machinery is the following. One start with the sequences in A and one prolong them to the right with the strings in B_d, B_u, acording to the sticker operation (the elements of B_d are used on the lower row, and those of B_u are used on the upper row). When no blank symbol is present, we obtain a string over the alphabet $\begin{pmatrix} V \\ V \end{pmatrix}_\rho$. The language of all such strings is the language generated by γ.

Formally, [3] defines this language as follows:
For two strings x, z ∈ $W_\rho(V)$ one write x⇒z iff z = μ(x,y) for some y ∈ $B_d \cup B_u$.

One denote by ⇒* the reflexive and transitive closure of the relation ⇒, like usually.

A sequence $x_1 \Rightarrow x_2 \Rightarrow \ldots \Rightarrow x_k$, $x_1 \in A$ is called a computation in γ (of length k – 1). A computation as above is *complete* if $x_k \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^*$ (no blank symbol is present in the last string of composite symbols).

The language generated by γ, denoted by L(γ), is defined by

$$L(\gamma) = \left\{ w \in \begin{pmatrix} V \\ V \end{pmatrix}_\rho^* \mid x \Rightarrow^* w,\ x \in A \right\}.$$

Therefore, only the *complete* computation are taken into account when defining L(γ).

3. Resolution method as a sticker system

One of the most largely used refutation method in automated theorem proving is resolution method. It can be introduced as a formal system [5]:

$$R = (\Sigma_R, F_R, A_R, R_R)$$

where:

Σ_R = {p, q, r,...,p_i, q_i, r_i} ∪ {⌉,∨}. p, q, r, ...p_i, q_i, r_i are propositional variables.

F_R is a set {f, g,...,f_i, g_i,...} of clauses and they have the following form:

$$p_{i_1}^{\alpha_1} \vee p_{i_2}^{\alpha_2} \vee \ldots \vee p_{i_k}^{\alpha_k}$$

where

$$p^\alpha = \{ \begin{matrix} p & if\ \alpha = 1 \\ \overline{p} & if\ \alpha = 0 \end{matrix}$$

The constructs p or $\overline{p}$ are called literals. A special clause is the empty clause which is free from literals. It is noted as .

A_R = Φ. (The resolution system has no axioms.)

R_R = {*res*}, so the only deduction rule is called *res*, and it is defined as follows:

$$f \vee p,\ g \vee \quad \vdash f \vee g \qquad f, g \in F_R$$

It is known that any problem as: u_1, ..., u_2 $\vdash$ v is equivalent with $\{u_1,\ldots,u_n,\overline{v}\}$ unsatisfiable. Moreover each formula $u_1,\ldots,u_n,\overline{v}$ can be reduced as a set of clauses. So, to verify u_1, ..., u_2 $\vdash$ v is equivalent with to verify if a set of clauses is unsatisfiable.

Theorem (J.A. Robinson)

A set of clauses from propositional computation is unsatisfiable (or contradictious) if and only if C $\vdash$.

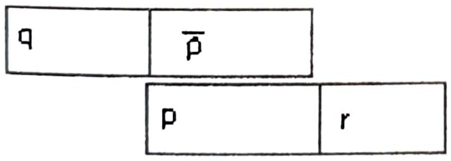
What we will explain here is applicable if original disjunctions have the length at most 2 (contain at most two variable). In 4 we'll show that any refutation by resolution can be reduced at the clauses with the length at most 2.

We codify each clause of C by a single stranded DNA formed by the sequence of the variables in that clause. For each clause p∨q we put in the test tube two kind of sequences: pq and qp.

Due to the complementarity, through the application of the resolution rule between two DNA sequences it will be fonned a new DNA sequence of higher length.

Example:

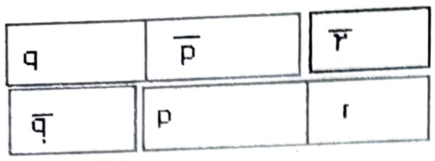
$$C = \{q \vee \overline{p}, r \vee p\}$$



It is clear that the obtaining of the empty clause means a complete DNA sequence double stranded. In the terms of sticker systems, the deduction of the empty clause means a complete computation (no blank symbols is present in the last string of compoşite symbols). In the previous example, if we have also the $\overline{r}$ and $\overline{q}$ clauses, we can obtain the empty clauses, which means a DNA sequence without blank symbols:

Example:

$$C = \{q \vee \overline{p}, r \vee p, \overline{q}, \overline{r}\}$$



Definition

The sticker system asociated with a set C of clauses is:

$$\gamma = (V, \rho, A, B_d, B_u),$$

where :

- V is the set of propositional variables in clauses,
- is the complementarity relation in propositional calculus,
- B_d is a set of elements of $\begin{pmatrix} \# \\ V \end{pmatrix}^*$, constructed as:

$$b_d \in B_d,\ b_d = \begin{pmatrix} \# \\ p_{i_1}^{\alpha_1} \end{pmatrix}\begin{pmatrix} \# \\ p_{i_2}^{\alpha_2} \end{pmatrix}\cdots\begin{pmatrix} \# \\ p_{i_n}^{\alpha_n} \end{pmatrix}$$

if $p_{i_1}^{\alpha_1} \vee p_{i_2}^{\alpha_2} \vee \ldots \vee p_{i_k}^{\alpha_k} \in C$

B_u is a set of elements of $\begin{pmatrix} V \\ \# \end{pmatrix}^*$, constructed as:

$$b_u \in B_u,\ b_u = \begin{pmatrix} p_{i_1}^{\alpha_1} \\ \# \end{pmatrix}\begin{pmatrix} p_{i_2}^{\alpha_2} \\ \# \end{pmatrix}\cdots\begin{pmatrix} p_{i_n}^{\alpha_n} \\ \# \end{pmatrix}$$

if $p_{i_1}^{\alpha_1} \vee p_{i_2}^{\alpha_2} \vee \ldots \vee p_{i_k}^{\alpha_k} \in C$

$A = B_d \cup B_u$.

(Let us remark that $k \leq 2$, see section 4).

In this sticker system, for two strings x, z $\in W_\rho(V)$ we have:

$x \Rightarrow z$ (or $z = \mu(x,y)$ for some $y \in B_d \cup B_u$)

if x is a string, y is a string from B_d or B_u (a clause) and z is a string obtained by resolution between y and x (more exactly, between y and the suffix of x).

A part of the Robinson's theorem can now be stated as:

Theorem

A set of clauses C, with at most two variables, is unsatisfiable if $L(\gamma) \neq \phi$, where γ is the sticker system associated with C.

4. Propositional calculus considerations.

In this section we will justify on limitation at clauses with at most two literals and the formulation as a sufficient conditions, of Robinson's theorem, in terms of sticker systems.

Let us consider the four posibilities for a formula:

a) with $\wedge$ in conclusion

b) with $\vee$ in premise

c) with $\vee$ in conclusion

d) with $\wedge$ in premise.

a) For this case, we consider the theorem:

$$\vdash (p\to q)\wedge(p\to r)\leftrightarrow(p\to q\wedge r)$$

This theorem says that, to prove a theorem as "$p\to q\wedge r$" is enough to prove "$p\to q$" and "$p\to r$". Both formulae, "$p\to q$" and "$p\to r$" ,(or, as clauses $\overline{p}\vee q,\ p\vee\overline{r},\ldots$) have a number of literals less than "$p\to q\wedge r$".

b) For this case let us consider the theorem

$$\vdash (p\to r)\wedge(q\to r)\leftrightarrow(p\vee q\to r)$$

so, to prove a theorem as "$p\vee q\to r$" is enough to prove "$p\to r$" and "$q\to r$".

Both these formulae have a number of literals less than "$p\vee q\to r$".

The cases a) and b) provide the following observation:

If one of formulae from the set $\{u_1,\ldots,u_n,\overline{v}\}$, let say u_i, is of the form:

$$p_{i_1}^{\alpha_1}\vee p_{i_2}^{\alpha_2}\vee\ldots\vee p_{i_k}^{\alpha_k}\to q_{j_1}^{\beta_1}\wedge q_{j_2}^{\beta_2}\wedge\ldots\wedge q_{j_l}^{\beta_l}$$

then this formulae will introduce k·l clauses with two literals of the form:

$$\overline{p_{i_1}^{\alpha_1}}\vee q_{j_1}^{\beta_1},\overline{p_{i_2}^{\alpha_2}}\vee q_{j_2}^{\beta_2},\ldots,\overline{p_{i_k}^{\alpha_k}}\vee q_{j_k}^{\beta_k}.$$

If the set of these clauses (for u_i) in union with the rest of clauses (for $u_1,\ldots,u_{i-1},u_{i+1},\ldots,\ \overline{v}$) is unsatisfiable, then we conclude that $\{u_1,\ldots,u_n,\overline{v}\}$ is unsatisfiable AND conversely.

c) For this case we have the implication:

$$\vdash (p\to q)\wedge(p\to r)\to(p\to q\vee r)$$

Let us observe that the reverse implication is not a valid formula. As clauses, this formula can be rewriten as:

$$\vdash(\overline{p}\vee q)\wedge(\overline{p}\vee r)\to(\overline{p}\vee q\vee r)$$

By the equivalence:

$$\vdash (u \to F) \wedge (v \to F) \leftrightarrow (u \vee v \to F)$$

$$(\text{or } \vdash \overline{u} \wedge \overline{v} \leftrightarrow \overline{u \vee v})$$

we can deduce that if a set of clauses:

$$C = \{\overline{p} \vee q, \overline{p} \vee r, \ldots\}$$

is unsatisfiable, then the set of clauses

$$C' = \{\overline{p} \vee q \vee r, \ldots\}$$

is also unsatisfiable.

d) In this case we have the implication:

$$\vdash (p \to r) \wedge (q \wedge r) \to (p \wedge q \to r)$$

or

$$(\overline{p} \vee r) \wedge (\overline{q} \vee r) \to (\overline{p} \vee \overline{q} \vee r) \vdash$$

Similar with the case c), we can deduce that if a set of clauses:

$$C = \{\overline{p} \vee r, \overline{q} \vee r, \ldots\}$$

is unsatisfiable, then the set of causes

$$C' = \{\overline{p} \vee \overline{q} \vee r, \ldots\}$$

is also unsatisfiable.

The cases c) and d) provide the following observation:

If one of formulae $\{u_1, \ldots, u_n, \overline{v}\}$, let say u_i, is the form:

$$p_{i_1}^{\alpha_1} \wedge p_{i_2}^{\alpha_2} \wedge \ldots \wedge p_{i_k}^{\alpha_k} \to q_{j_1}^{\beta_1} \vee q_{j_2}^{\beta_2} \vee \ldots \vee q_{j_l}^{\beta_l}$$

then this formula will introduce k•l clauses with two literals of the form:

$$\overline{p_{i_1}^{\alpha_1}} \vee q_{j_1}^{\beta_1}, \overline{p_{i_2}^{\alpha_2}} \vee q_{j_2}^{\beta_2}, \ldots, \overline{p_{i_k}^{\alpha_k}} \vee q_{j_k}^{\beta_k}$$

If the set of these clauses (for u_i) in union with the set of clauses for

$$u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_n, \overline{v}$$

is unsatisfiable, then we conclude that $\{u_1, \ldots, u_n, \overline{v}\}$ is unsatisfiable (but not AND conversely).

REFERENCES

1. L. Adleman, *Molecular Computation of Solutions to Combinatorial Problems*, Science, Vol 266, 1994, pp 1021.
2. Jacques Cohen: *Computational Molecular Biology: a promissing application using LP and its extensions.*The LP paradigms, a 25-year perspective, ed. K. Apt, Springer, 1999.
3. L. Kari, Gh Păun, G. Rozenberg, A. Salomaa, Sheng Yu, *DNA computing, sticker systems, and universality*, Acta Informatica 35, 1998, pp 401-420.
4. Gh. Păun: *Splicing, A challenge for formal language theorist*, Buletin of EATCS, nr 57, 1995, pp183-194.

5. D. Tătar: *The mathematical bases of computer science*, Univ Babeş-Bolyai, Cluj-Napoca, 1993 (in romanian).

University " Babes- Bolyai", Cluj-Napoca, Romania
Faculty of Mathematics and Computer Science
Department of Computer Science

Email address: dtatar@cs.ubbcluj.ro, moltean@cs.ubbcluj.ro