A Deductive Method for Construction and Visualization of Contigs in the STS Strategy

Masami Hagiya
hagiya@is.s.u-tokyo.ac.jp

1 Department of Information Science
Graduate School of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113 Japan

Abstract

The problem of constructing contigs by the STS strategy is a simple combinatorial problem if the given hit information is correct and complete. However, hit information is often incorrect or incomplete due to failure or inability of experiments. Moreover, in addition to hit information, various sources of information are also available, such as known landmarks, other clone libraries, etc. In order to cope with incompleteness, incorrectness and additional information, we developed a deductive method for constructing contigs. Contigs are constructed by deducing an equivalence relation of clone directions and a partial order among STS markers on each equivalence class of directions. In the paper, a practical algorithm based on the method is presented and its completeness is proved. The method is also axiomatized by a set of inference rules for deducing the equivalence relation and the partial orders. We finally discuss the problem of visualizing contigs based on the information deduced by our method.

1 Introduction

A sequence-tagged site (STS) is a short DNA sequence that is uniquely positioned in the whole genome, and can be used as a marker in the genome. Polymerase chain reaction (PCR) is an efficient method for detecting whether a clone contains the given STS marker or not. If two clones are found to contain the same STS marker, then those clones must overlap. By using this overlapping information, one can construct contiguous units (contigs) of clones. This is called the STS strategy for constructing contigs [2, 1, 3, 6].

In principle, STS markers can be made from any part of the genome. For a set of unordered clones, however, the strategy usually taken is to make an STS marker from each end of a clone.

1 松谷昌己, 東大理情報, 〒113 文京区本郷7-3-1
Thus, two STS markers correspond to each clone. In addition to the markers made from a clone’s ends, one can also use STS markers published in the literature, whose position or order in the genome is already known. We call such STS markers landmarks in this paper.

If an STS marker is contained in a clone, we say that the marker hits the clone. The information of whether each marker hits a clone or not is called hit information in this paper. The problem of constructing contigs by the above STS strategy is a simple combinatorial problem, if the hit information obtained by experiments is correct and complete, i.e., if it is correctly and completely known whether a marker hits a clone or not. However, hit information is often incorrect or incomplete due to failure and inability of experiments. For example, if an end of a clone is found to be a repetitive sequence, then that end cannot be used as an STS marker. Moreover, in addition to hit information, various sources of information are also available, such as known landmarks or other clone libraries. For example, one can order clones using the known order among landmarks.

In order to cope with incompleteness, incorrectness and additional information, we developed a deductive method for constructing contigs. Contigs are constructed by deducing an equivalence relation of clone directions and a partial order among STS markers on each equivalence class of directions. This method is clear and easy to understand, because it is based on well-established mathematical concepts.

We derived a practical algorithm based on the method. The algorithm is actually used in a computer software called ContigMaker, which is being developed at Institute for Medical Science of University of Tokyo [5]. The software is designed to support database management of clones and markers, construction of contigs, and visualization of clone maps.

The soundness of the algorithm is trivial, but its completeness is not. We proved that the algorithm is complete in the sense that if two markers cannot be ordered by the algorithm, then the order between the two markers can be reversed without causing inconsistency with hit information.

A rough proof of the completeness is included in the paper. The proof clarifies the situation where markers cannot be ordered by the algorithm. This is because a small contig is completely contained in a large contig and the direction of the small one cannot be determined with respect to the direction of the large one. Notice that the method of this paper does not assume any restriction on relative sizes of clones. A clone can be completely contained in another clone.

The method is then axiomatized by a set of inference rules for deducing the equivalence relation and the partial orders. The inference rules are formalized as definite clauses in logic programming. Therefore, it is easy to use additional information by adding new definite clauses to the axiomatization.

Unfortunately, with the current technology of deductive databases [4], it is still inefficient to directly execute the axiomatization. We can, however, extend the practical algorithm according to the new definite clauses formalizing the additional information. Formalization by definite clauses clarifies the additional information and helps to extend the algorithm.

At the end of the paper, we discuss the problem of visualizing contigs. When markers are drawn in a contig map, a total order is enforced on the markers even if they are not ordered by the algorithm. This means that a contig map contains more information than logically implied by hit information. Therefore, when a marker is drawn to the left of another marker, one cannot judge whether the order is a logical consequence or is just a result of visualization. In this paper, we propose some new ideas for distinguishing the information logically implied by
hit information from the information that is not. We also discuss the possibility of allowing to edit contig maps.

The rest of the paper is organized as follows. In Section 2, we describe the practical algorithm for contig construction based on our method. In Section 3, the completeness of the algorithm is shown. In Section 4, the method is axiomatized by definite clauses. In Section 5, visualization of contigs is discussed.

2 Practical Algorithm

We assume that two STS markers are made from two ends of each clone. Those markers are tentatively called left and right ends of the clone, and denoted by \( \text{left}(C) \) and \( \text{right}(C) \) for clone \( C \). We also assume in this section that there are no known landmarks available, i.e., we assume that a marker is always of the form \( \text{left}(C) \) or of the form \( \text{right}(C) \).

If marker \( M \) hits clone \( C \), we write \( \text{hit}(M, C) \). If marker \( M \) does not hit clone \( C \), we write \( \text{not}\_\text{hit}(M, C) \). One can define \( \text{not}\_\text{hit}(M, C) \) to be the negation of \( \text{hit}(M, C) \) if hit information is complete.

Let \( C_1 \) and \( C_2 \) be two clones. If the direction from \( \text{left}(C_1) \) to \( \text{right}(C_1) \) is identical to that from \( \text{left}(C_2) \) to \( \text{right}(C_2) \), we write \( C_1 \equiv C_2 \). If they are opposite to each other, we write \( C_1 \equiv -C_2 \) or \( -C_1 \equiv C_2 \). The relation \( \equiv \) is an equivalence relation (i.e., reflexive, symmetric and transitive relation), and the operator \( - \) is congruent with the equivalence relation and satisfies \(-(-C) \equiv C \) for any clone \( C \).

Let \( M_1 \) and \( M_2 \) be two markers. We write \( M_1 <_C M_2 \) if \( M_1 \) is to the left of \( M_2 \) in the direction of \( C \), i.e., the direction from \( M_1 \) to \( M_2 \) is identical to that from \( \text{left}(C) \) to \( \text{right}(C) \).

The practical algorithm works in two stages. In the first stage, the equivalence relation of clone directions is computed. The only rule used in this stage is that if the situation in Figure 1 is found, then \( C_1 \equiv C_2 \) is derived, and if the situation in Figure 2 or Figure 3 is found, then \( C_1 \equiv -C_2 \) is derived. The algorithm then computes the reflexive, symmetric and transitive closure of the facts derived by this rule. It also refers to the properties of the operator \(-\).

If the result of the first stage consists of only two equivalence classes, i.e., if \( C_1 \equiv C_2 \) or \( C_1 \equiv -C_2 \) holds for any pair of clones \( C_1 \) and \( C_2 \), then it means that the directions of all the clones have been adjusted and there exists only a single contig.

In the second stage of the algorithm, the relation \( <_C \) is computed. According to the meaning of the relation, it has the following properties.

- If \( C \equiv C' \) and \( M_1 <_C M_2 \), then \( M_1 <_{C'} M_2 \).
- If \( M_1 <_C M_2 \), then \( M_2 <_{-C} M_1 \).
left($C_1$) \quad right($C_1$)

hit(right($C_2$), $C_1$) \quad \text{hit(right($C_1$), $C_2$)}

right($C_2$) \quad \text{left($C_2$)}

Figure 2: $C_1 \equiv -C_2$

right($C_1$) \quad \text{left($C_1$)}

hit(left($C_2$), $C_1$) \quad \text{hit(left($C_1$), $C_2$)}

left($C_2$) \quad \text{right($C_2$)}

Figure 3: $C_1 \equiv -C_2$

- If $M_1 <_C M_2$ and $M_2 <_C M_3$, then $M_1 <_C M_3$.

The last property says that $<_C$ is a transitive relation.

The algorithm computes the relation $<_C$ by using the above properties together with the following rules.

- left($C$) $<_C$ right($C$) for any clone $C$.
- If hit($M$, $C$), then left($C$) $<_C$ $M$ $<_C$ right($C$).
- If not hit($M$, $C$) and $M <_C$ right($C$), then $M <_C$ left($C$).
- If not hit($M$, $C$) and left($C$) $<_C$ $M$, then right($C$) $<_C$ $M$.

The first rule is obvious from the definition of $<_C$. The second rule refers to hit information positively, and the last two rules refer to hit information negatively. Notice that the last two are conditional rules. The situations in which they are used are depicted in Figures 4 and 5.

If there are only two equivalence classes of clone directions at the end of the first stage of the algorithm, the second stage is reduced to computing a single transitive relation. Otherwise, there exist more than one relations to be computed in the second stage.

left($C$) \quad right($C$)

\begin{center}
\begin{tikzpicture}[->,>=stealth',shorten >=1pt,auto,transform shape]
  \node (M) {$M$};
  \node (left) at (M-|left) {$\text{left}(C)$};
  \node (right) at (M-|right) {$\text{right}(C)$};
  \draw (left) -- (right);
  \end{tikzpicture}
\end{center}

Figure 4: not hit($M$, $C$)
3 Completeness of the Algorithm

The algorithm is obviously sound, i.e., if \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is derived by the algorithm, then \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is true.

Referring to the notions of provability and truth in logic, let us write \( \vdash C_1 \equiv C_2 \) or \( \vdash M_1 <_C M_2 \) if \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is derivable by the algorithm, and write \( \models C_1 \equiv C_2 \) or \( \models M_1 <_C M_2 \) if \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is true.

To be more precise, we have to be explicit about the notion of a model. A model is a total order among the markers that is compatible with the given hit information. Let \( M \) be a model. We write \( M \models C_1 \equiv C_2 \) or \( M \models M_1 <_C M_2 \) if \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is true under \( M \).

The soundness of the algorithm means that if \( \vdash C_1 \equiv C_2 \) then \( M \models C_1 \equiv C_2 \), and if \( \vdash M_1 <_C M_2 \) then \( M \models M_1 <_C M_2 \) for any model \( M \).

If the given hit information is complete, the above algorithm is also complete in the sense that if \( C_1 \equiv C_2 \) or \( M_1 <_C M_2 \) is not derivable by the algorithm, then there is a possibility that \( C_1 \equiv -C_2 \) or \( M_2 <_C M_1 \) is true, i.e., there exists some model \( M \) such that \( M \models C_1 \equiv -C_2 \) or \( M \models M_2 <_C M_1 \) holds.

To prove the completeness of the algorithm, we first make some definitions. Let \( \text{contig}(C) \) denote a set of clones as follows.

\[
\text{contig}(C) = \{C' \mid \vdash C \equiv C' \text{ or } \vdash C \equiv -C'\}
\]

If \( S \) is a set of clones, \( \text{marker}(S) \) denotes the set of markers made from clones in \( S \).

\[
\text{marker}(S) = \{\text{left}(C) \mid C \in S\} \cup \{\text{right}(C) \mid C \in S\}
\]

We now state the most important lemma needed to prove the completeness.

**Lemma 1.** The set \( \text{marker}(\text{contig}(C)) \) of markers is totally ordered with respect to \( <_C \) by the algorithm, i.e., if \( M_1, M_2 \in \text{marker}(\text{contig}(C)) \) and \( M_1 \neq M_2 \), then \( \vdash M_1 <_C M_2 \) or \( \vdash M_2 <_C M_1 \).

**Proof sketch** This lemma is proved by induction on the construction of \( \text{contig}(C) \). We first order clones in \( \text{contig}(C) \) in such a way that \( \text{contig}(C) = \{C_1, ..., C_n\} \), and for each \( i, C_i \) is related to one of \( C_1, ..., C_{i-1} \) by one of the configurations in Figures 1-3.

We then prove by induction on \( i \) that the set \( \text{marker}([C_1, ..., C_i]) \) of markers is totally ordered with respect to \( <_C \) by the algorithm. (end of proof sketch)

Two markers \( M_1 \) and \( M_2 \) in \( \text{marker}(\text{contig}(C)) \) are called adjacent if \( \vdash M_1 <_C M_2 \) and there exists no marker \( M \) in \( \text{marker}(\text{contig}(C)) \) such that \( M \models M_1 <_C M <_C M_2 \). We say
that marker $M$ hits a set $S$ of clones if $M$ hits some clone in $S$. If $M$ hits $\text{contig}(C)$ and $\vdash M_1 \prec C \ M \prec C \ M_2$, where $M_1$ and $M_2$ are adjacent markers in $\text{marker}(\text{contig}(C))$, we say that $M$ is in between $M_1$ and $M_2$.

Given the above lemma, the following lemma is easy to prove by case analysis.

**Lemma 2.** If $\text{left}(C')$ hits $\text{contig}(C)$ and $C'$ does not belong to $\text{contig}(C)$, then $\text{right}(C')$ also hits $\text{contig}(C)$. Moreover, there exist some adjacent markers $M_1$ and $M_2$ in $\text{marker}(\text{contig}(C))$ such that $\text{left}(C')$ and $\text{right}(C')$ are both in between $M_1$ and $M_2$. If $\text{right}(C')$ hits $\text{contig}(C)$ and $C'$ does not belong to $\text{contig}(C)$, then $\text{left}(C')$ also hits $\text{contig}(C)$ and there exist adjacent markers $M_1$ and $M_2$ as above.

By the above lemma, we can prove the following lemma.

**Lemma 3.** If $C_1 \equiv C_2$ is not derivable by the algorithm and a marker in $\text{marker}(\text{contig}(C_1))$ hits $\text{contig}(C_2)$, then all the markers in $\text{marker}(\text{contig}(C_1))$ are in between some adjacent markers $M_1$ and $M_2$ in $\text{marker}(\text{contig}(C_2))$.

We now want to prove the completeness of the algorithm. Assume that $C_1 \equiv C_2$ is not derivable by the algorithm. Let $M$ be a model. If $M \models C_1 \equiv \neg C_2$, we are done. So, assume that $M \models C_1 \equiv C_2$. Since $C_1 \equiv C_2$ is not derivable, $C_1$ does not belong to $\text{contig}(C_2)$ and $C_2$ does not belong to $\text{contig}(C_1)$. There are three cases.

In the first case, $\text{left}(C_1)$ hits $C_2$. By Lemma 2, both $\text{left}(C_1)$ and $\text{right}(C_1)$ are in between adjacent markers $M_1$ and $M_2$ in $\text{marker}(\text{contig}(C))$. By reversing the order of all the markers in between $M_1$ and $M_2$, we can construct a model $M'$ from the model $M$ such that $M' \models C_1 \equiv \neg C_2$.

The case that $\text{left}(C_2)$ hits $C_1$ is handled similarly.

If $\text{left}(C_1)$ does not hit $C_2$ and $\text{left}(C_2)$ does not hit $C_1$, then we reverse the order of all the markers in $\text{contig}(C_1)$ and all the markers hitting $\text{contig}(C_1)$. The order between $\text{left}(C_1)$ and $\text{right}(C_1)$ is reversed but the order between $\text{left}(C_2)$ and $\text{right}(C_2)$ is not changed.

Assume now that $M_1 \prec C \ M_2$ is not derivable by the algorithm. Let $M$ be a model. If $M \models M_2 \prec C \ M_1$, we are done. So, assume that $M \models M_1 \prec C \ M_2$. There are two cases.

If both $M_1$ and $M_2$ hit $\text{contig}(C)$, they must be in between adjacent markers $N_1$ and $N_2$ in $\text{marker}(\text{contig}(C))$. By reversing the order of all the markers in between $N_1$ and $N_2$, we can construct a model $M'$ from the model $M$ such that $M' \models M_2 \prec C \ M_1$.

If $M_1$ or $M_2$ does not hit $\text{contig}(C)$, we can construct a model $M'$ from the model $M$ by reversing the order of all the markers in $\text{contig}(C)$ and all the markers hitting $\text{contig}(C)$. The order between $\text{left}(C)$ and $\text{right}(C)$ is reversed but the order between $M_1$ and $M_2$ is not changed.

## 4 Axiomatization

The method of the paper can be axiomatized by a set of definite clauses in logic programming. Let $o(M_1, M_2, M_3, M_4)$ denote a predicate meaning that the direction from $M_1$ to $M_2$ is identical to that from $M_3$ to $M_4$. By this definition, $C_1 \equiv C_2$ is equivalent to

$$o(\text{left}(C_1), \text{right}(C_1), \text{left}(C_2), \text{right}(C_2)),$$
and $M_1 < C < M_2$ is equivalent to

$$o(M_1, M_2, \text{left}(C), \text{right}(C)).$$

We can then give the following definite clauses for $o$, which cover the algorithm presented so far.

\begin{align*}
\text{clone}(C) & \rightarrow o(\text{left}(C), \text{right}(C), \text{left}(C), \text{right}(C)) \\
o(L_1, R_1, L_2, R_2) & \rightarrow o(L_2, R_2, L_1, R_1) \\
o(L_1, R_1, L_2, R_2) & \rightarrow o(R_1, L_1, R_2, L_2) \\
o(L_1, R_1, L_2, R_2), o(L_2, R_2, L_3, R_3) & \rightarrow o(R_1, L_1, R_3, L_3) \\
o(M_1, M_2, L, R), o(M_2, M_3, L, R) & \rightarrow o(M_1, M_3, L, R) \\
hit(M, C) & \rightarrow o(\text{left}(C), M, \text{left}(C), \text{right}(C)) \\
hit(M, C) & \rightarrow o(M, \text{right}(C), \text{left}(C), \text{right}(C)) \\
\text{not.hit}(M, C), o(M, \text{right}(C), \text{left}(C), \text{right}(C)) & \rightarrow o(M, \text{left}(C), \text{left}(C), \text{right}(C)) \\
\text{not.hit}(M, C), o(M, \text{left}(C), \text{right}(C), \text{left}(C)) & \rightarrow o(M, \text{right}(C), \text{right}(C), \text{left}(C))
\end{align*}

By axiomatizing the algorithm by a set of definite clauses, one can easily add new knowledge by preparing new definite clauses. For example, if landmark1, landmark2 and landmark3 are known to be in this order, it is only necessary to add the following clause.

$$o(\text{landmark1}, \text{landmark2}, \text{landmark2}, \text{landmark3})$$

If there are errors in experiments and hit information contains inconsistency, or if additional information is inconsistent with hit information, it is possible to deduce inconsistency from the set of clauses. In such cases, one can find the sources of inconsistency by tracing back its deduction.

As can be seen from the above set of definite clauses, the axiomatization is compact and powerful. However, with the current technology of deductive databases, it is still inefficient to directly execute the axiomatization. The ordinary magic set method is not applicable to the above set of definite clauses. In particular, the computation of transitive closures is the most serious problem.

We can, however, extend the practical algorithm according to the new definite clauses formalizing the new knowledge. Formalization by definite clauses clarifies the new knowledge and helps to extend the algorithm.

## 5 Visualization

The result of constructing contigs is displayed in a contig map, where clones and markers are drawn according to the orders obtained by the method. As discussed in the introduction, however, a contig map contains more information than logically implied by hit information,
because when a contig map is displayed, orders between markers and contigs that are not derivable by the algorithm are arbitrarily chosen by the display program. In this section, in order to distinguish the information logically implied by hit information from the information that is not, we propose two ideas that are based on our method.

One is to color clones according to the equivalence relation of directions. If \( C_1 \equiv C_2 \) or \( C_1 \equiv -C_2 \), then clones \( C_1 \) and \( C_2 \) are given the same color. Otherwise, they are colored differently (Figure 6). When one looks at a contig map, he or she can immediately realize that the relative direction of two contigs is not a logical consequence of hit information if they are colored in different colors.

The other idea is to prepare a special display mode, in which a pair of markers or contigs is pointed out, if the order of the pair is not logically implied by hit information and they can be swapped without causing inconsistency with hit information. For example, in Figure 7, it is pointed out that two contigs (the contig drawn by small white circles and the one by small black circles) contained in a larger one (the contig drawn by simple lines) can be swapped without causing inconsistency.

Since biologists have more knowledge than is formalized and provided to the algorithm for constructing contigs, it seems desirable to allow them to edit a contig map according to their biological knowledge. However, they should not change the map so that a contig map becomes inconsistent with hit information.

If the distinction between the information logically implied by hit information and the information that is not is always kept by the editing program, it is possible to immediately report an error message when a contig map is tried to be edited in an inconsistent way.

6 Conclusions

We have presented an algorithm for constructing contigs from hit information of whether each STS marker made from a clone's end hits another clone or not. The algorithm is clear and easy to understand because it is based on well established mathematical concepts. We proved the
completeness of the algorithm by analyzing the situation where the algorithm fails to derive the order between two markers.

We then axiomatized the algorithm by a set of definite clauses. By the axiomatization, one can easily add new knowledge to the algorithm by preparing new definite clauses. We still need more researches for directly executing the axiomatization.

We finally discussed the problem of visualizing contigs. We gave some ideas for faithfully visualizing the information obtained by our algorithm.

**Acknowledgments**

The author deeply thanks to all the members of the ContigMaker group. This work was supported by a Grand-in-Aid for Scientific Research on Priority Areas, 'Genome Informatics', from the Ministry of Education, Science and Culture of Japan.

**References**


