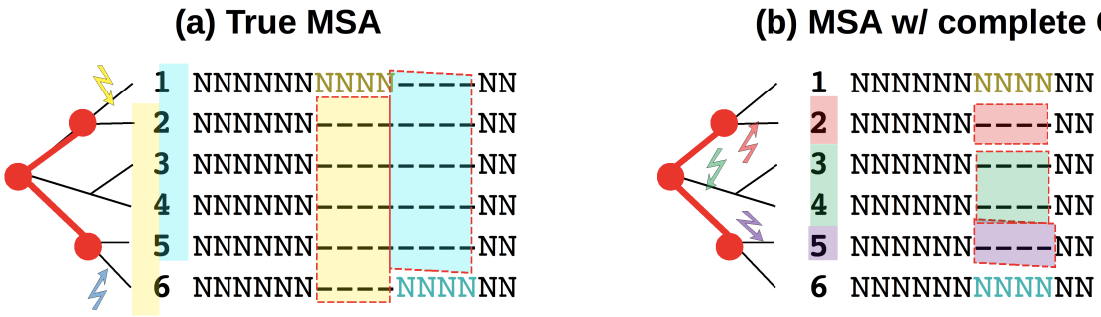
Contemplating on an Algorithm to “Reverse” a(n) (i)CII ((incomplete) collapse of independent insertions)

By Kiyoshi Ezawa
From: Sep 20, 2019,
Till: Sep 22, 2019

(1-i) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (i).



(1-i) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (i).

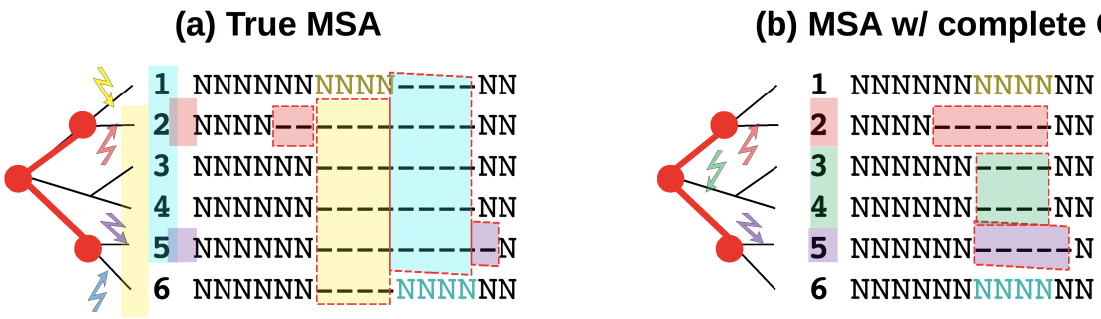
When two (or more) independent "insertion"s (yellow & cyan) in the true MSA **(a)** collapse completely,
the resulting MSA **(b)** should typically contain two or more "deletion"s (red, green & purple) that are horizontally overlapping but NOT interfering with each other.

In each panel, the gap-blocks are separated from each other with the red nodes and edges.

(1) A notable point is that the **"neighboring" non-interfering gap-blocks** in **panel b** are intervened by (either) TWO tri-valent nodes plus an edge (or one higher-valent node).

(2) Another notable point is that there are at least two overlapping gap-blocks (in **panel b**), each of which has NO "sibling" gap-block that horizontally overlaps with it (in the region under consideration).

(1-ii) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (ii).



(1-ii) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (ii).

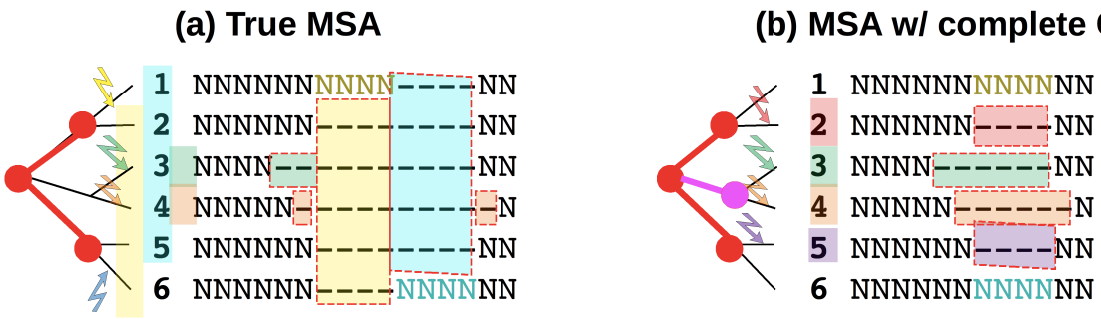
It should be noted that

the resulting non-interfering horizontally overlapping "deletion"s (red, green & purple (in **panel b**)) may overlap each other INcompletely;

In such a case, the true MSA **(panel a)** should contain (a) shorter "deletion"(s) (red & purple in this case).

The notable points apply to this case, as well.

(1-iii) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (iii).



(1-iii) To Detect a candidate (complete) CII, let's examine the characteristic features of c-CII (iii).

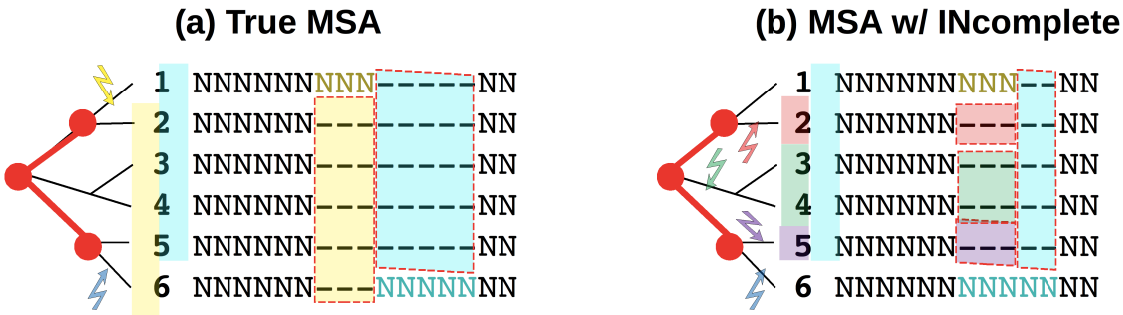
In some cases,

some of the resulting horizontally overlapping "deletion"s (in this case, green & (in **panel b**)) may be "siblings" to each other;

In such a case, too, the true MSA (**panel a**) should contain some shorter "deletion"s (green & in this case).

The notable points apply to this case, as well,
PROVIDED THAT the overlapping **"sibling" gap-blocks** are lumped together to give a **"parent" gap-block**, which will replace the "siblings" in the set of overlapping "deletion"s.

(2-i) To Detect a candidate (INcomplete) CII, let's examine the characteristic features of i-CII (i).



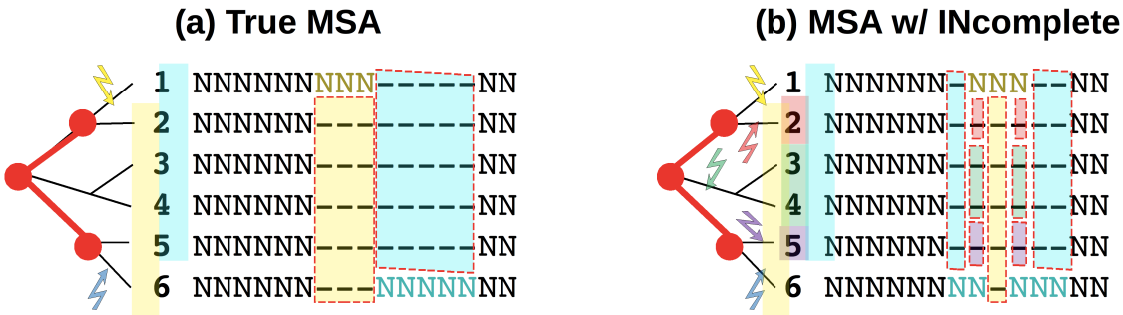
(2-i) To Detect a candidate (INcomplete) CII, let's examine the characteristic features of i-CII (i).

When independent "insertion"s (yellow & cyan) in the true MSA **(a)** collapse INcompletely,
the resulting MSA **(b)** should typically contain (a) left-over part(s) of the collapsed "insertion"(s) (cyan this time), in addition to two or more "deletion"s (red, green & purple) that are horizontally overlapping but NOT interfering with each other (as resulting from a complete CII).

The notable points apply to this case, as well,
IF we focus on the "deletion"s.

* The argument extends also to the situations with IN-completely overlapping "deletion"s, as in (1-ii) and (1-iii).

(2-ii) To Detect a candidate (INcomplete) CII, let's examine the characteristic features of i-CII (ii).



(2-ii) To Detect a candidate (INcomplete) CII, let's examine the characteristic features of i-CII (ii).

In (2-i), the left-over part of the collapsed "insertion"s was *very simple*, representing only a single (shorter) "insertion".

In real life, however, we may encounter **more complex situations**, as exemplified here.

Here in (2-ii), the left-over parts occur at multiple positions and on BOTH "insertion"s.

EVEN with such a complex MSA, the problem should be handled just as in the simple case,

PROVIDED THAT we f**irst focus only on the "deletion"s**;

Just ***merge*** the left-over "insertions" to the resulting "insertions" **AFTER** removing the spurious "deletion"s.

(This can be done because the "deletion"s are vertically included in BOTH the "insertion"s.)

The notable points apply to this case, as well,
IF we focus on the "deletion"s.

* The argument extends also to the situations with IN-completely overlapping "deletion"s, as in (1-ii) and (1-iii).

(3) An Algorithm to detect candidates of (c/i)-CIIs (1/2)

1. At each column, identify the **set** of MSA rows with the **"presence"** state; if the set is full, empty, or monophyletic (i.e., delimited by a single branch), mark it as "F", "E", or "Mono", respectively; otherwise, mark it as **"CII-cand"**;
2. ***Cluster*** neighboring columns with the same "presence" set, to form a **cluster** of columns;
3. ***Merge*** clusters with the same "presence" set (marked as "CII-cand"), if the "presence" set(s) of all mediating columns (or cluster(s)) is/are included in that of the subject clusters; The maximum region spanned by the clusters thus merged (as well as the mediating columns (or cluster(s))) will define a **"CII-candidate region"**;
4. For each "CII-candidate region", the "presence" set will be *divided into* **maximum monophyletic sets**, as follows (see next slide):

(3) An Algorithm to detect candidates of (c/i)-CIIs (2/2)

i. Examine whether the current "presence" set is delimited by a single branch; if so, put the branch into the output list, and end the algorithm;

ii. Pick an MSA row from the "presence" set, and identify the "subject" branch delimiting the row;

iii. If the branch has an effective parent branch, get a set of rows delimited by the "parent"; Otherwise, (A) remove the set of rows delimited by the subject branch from the current "presence" set, put the subject branch (actually, its uniquely chosen equivalent) into the output list, and return to (i);

iv. Examine whether the set of rows delimited by the "parent" branch is included in the current "presence" set; IF SO, make the "parent" be the "subject" branch, and return to (iii); Otherwise, perform (A) in (iii);

v. Thus, each **"CII-candidate" region** is defined with the **left- and right-end columns**, and the set of **maximum monophyletic sets of "presence" rows (and their** (uniquely chosen) **delimiting branches)**.

(5) An Algorithm to "reverse" the (c/i)-CII. (1/2)

- Input: A **"CII-candidate" region**, defined with the left- and right-end columns, and a set of maximum monophyletic sets of "presence" rows; A **set of columns in the input MSA**, each of whose cell is occupied by either the site (or residue) number of each sequence or a gap;

1. Prepare a set of **new columns** for each maximum monophyletic set; each set is initialized as empty;
2. From the left-end to the right-end of the region, "*(vertically) split*" each column into **"new" columns,** each of which inherits the sites corresponding to its *associated* maximum monophyletic set of rows, and fills the remaining rows with gaps; if the "new" column is null, discard it; otherwise, append it into the corresponding set of new columns;

(5) An Algorithm to "reverse" the (c/i)-CII. (2/2)

3. Create an output MSA (actually the set of its columns) by merging (1) input columns on the left, (2) the sets of new columns created as in 1 & 2, (3) input columns on the right; (it would be better to remove **null columns**, if any);
4. If a **resulting set** and/or (a) **flanking set**(s) of columns have the same (uniquely defined) delimiting branch, make them contiguous by altering the order among the new component sets; (it would be better to DEFINE the order according to the branch IDs);
5. Regarding the output MSA, compute the new set of gap-blocks, etc. by returning to their basic definitions (i.e., by performing the subroutines intended for the input MSA).

   NOTE: In the future, also modify the **"purge"-candidate regions** so that they will reflect the changes caused by "reverse"ing the CII.