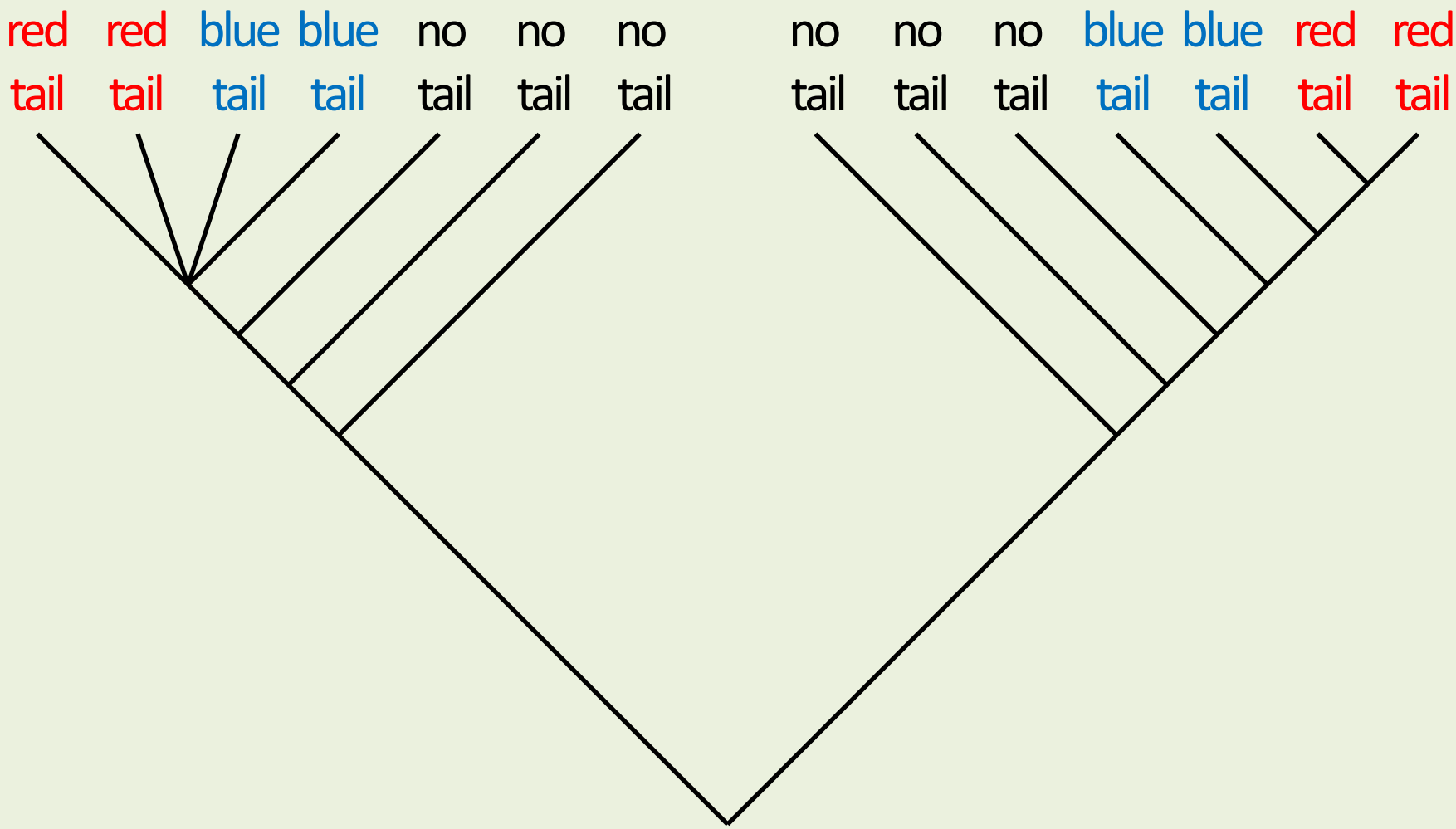
Theory and practice of parsimony analysis when some characters are inapplicable in some terminals.

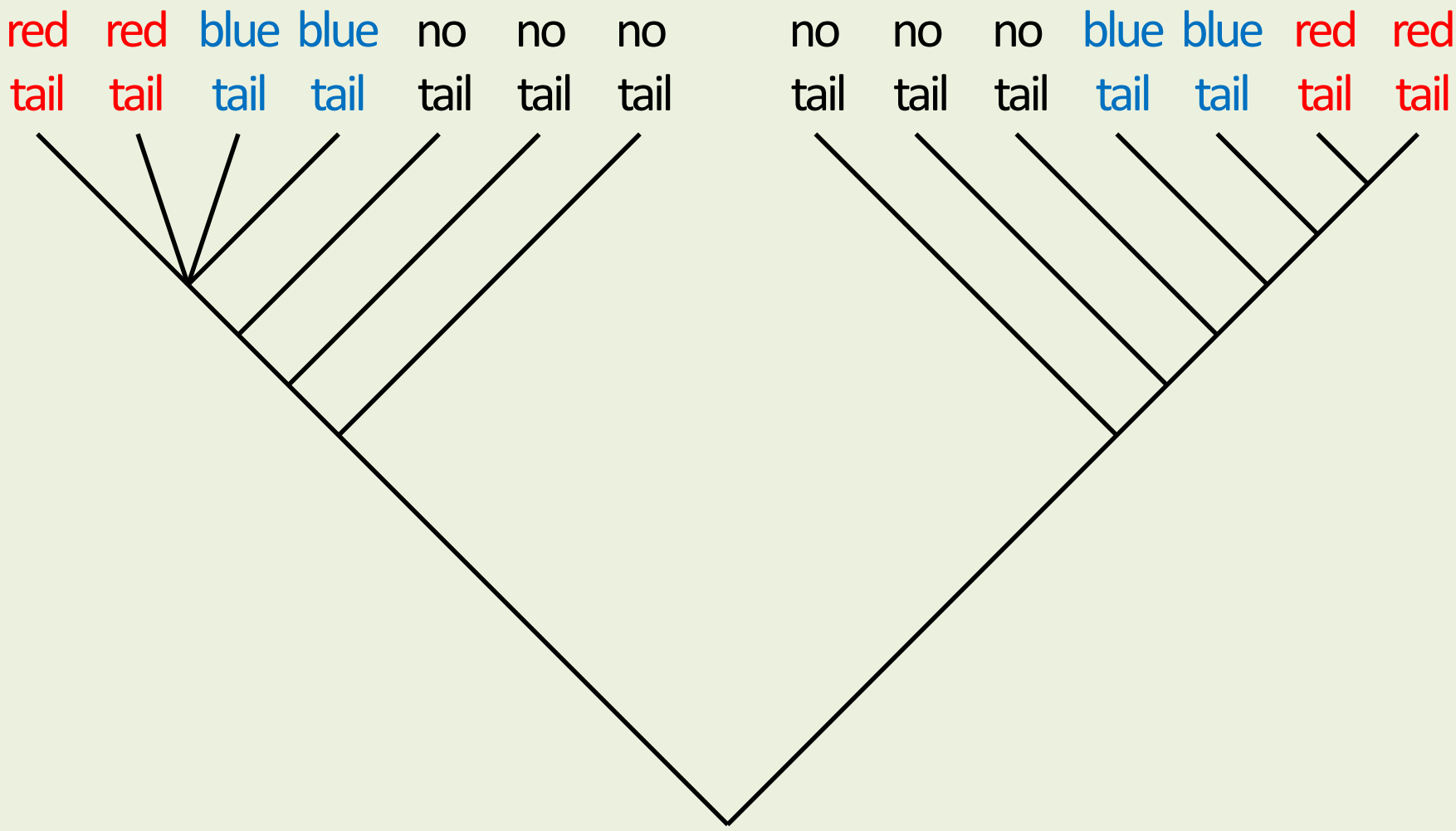
Jan De Laet

Gothenberg Botanical Garden

This is a pdf version of DOI 10.13140/RG.2.2.12159.51363
In this version, duplicate slides are not shown

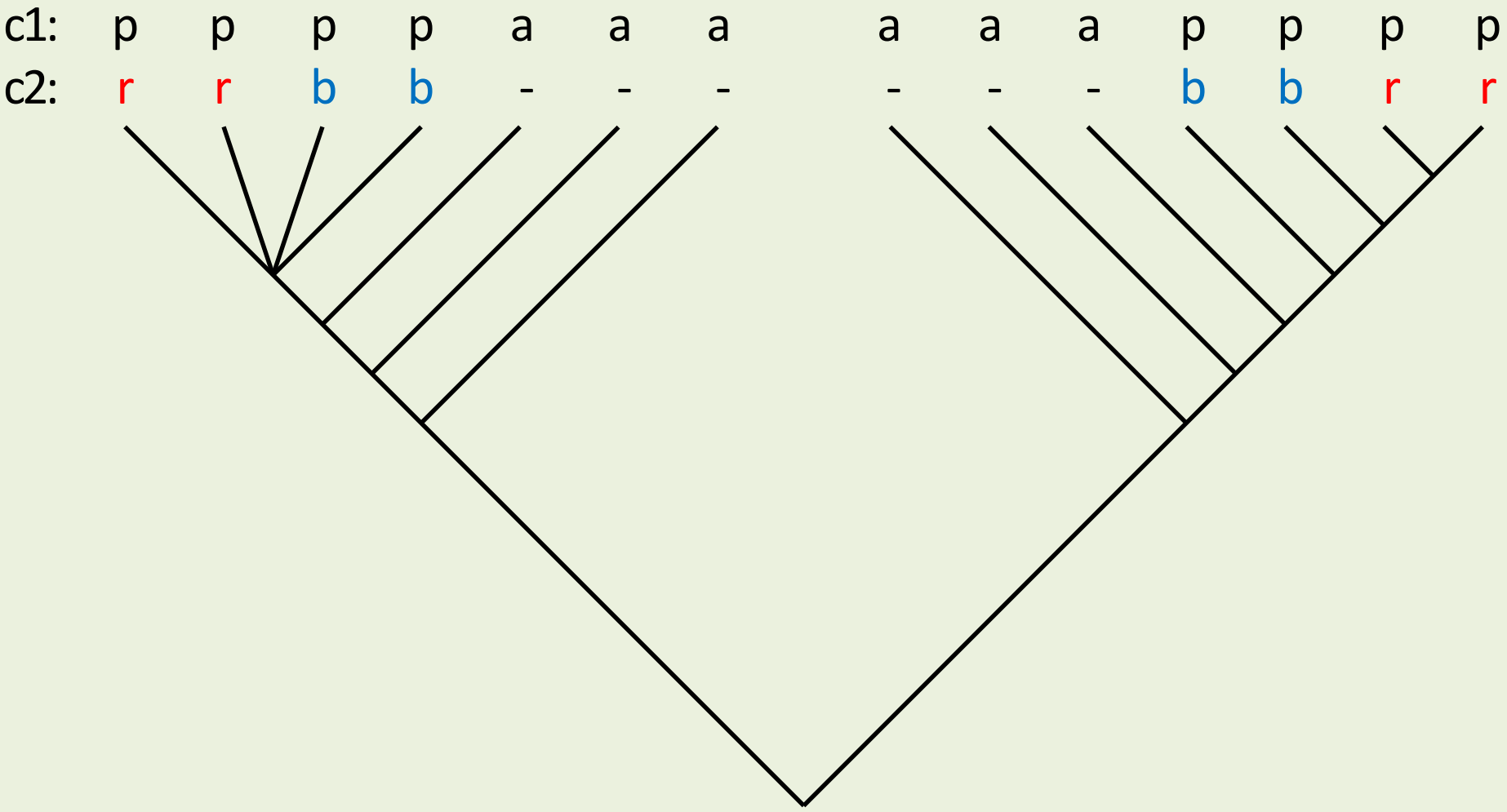


Maddison W.P. 1993. Missing data versus missing characters in phylogenetic analysis. Systematic Biology 42: 576-581.



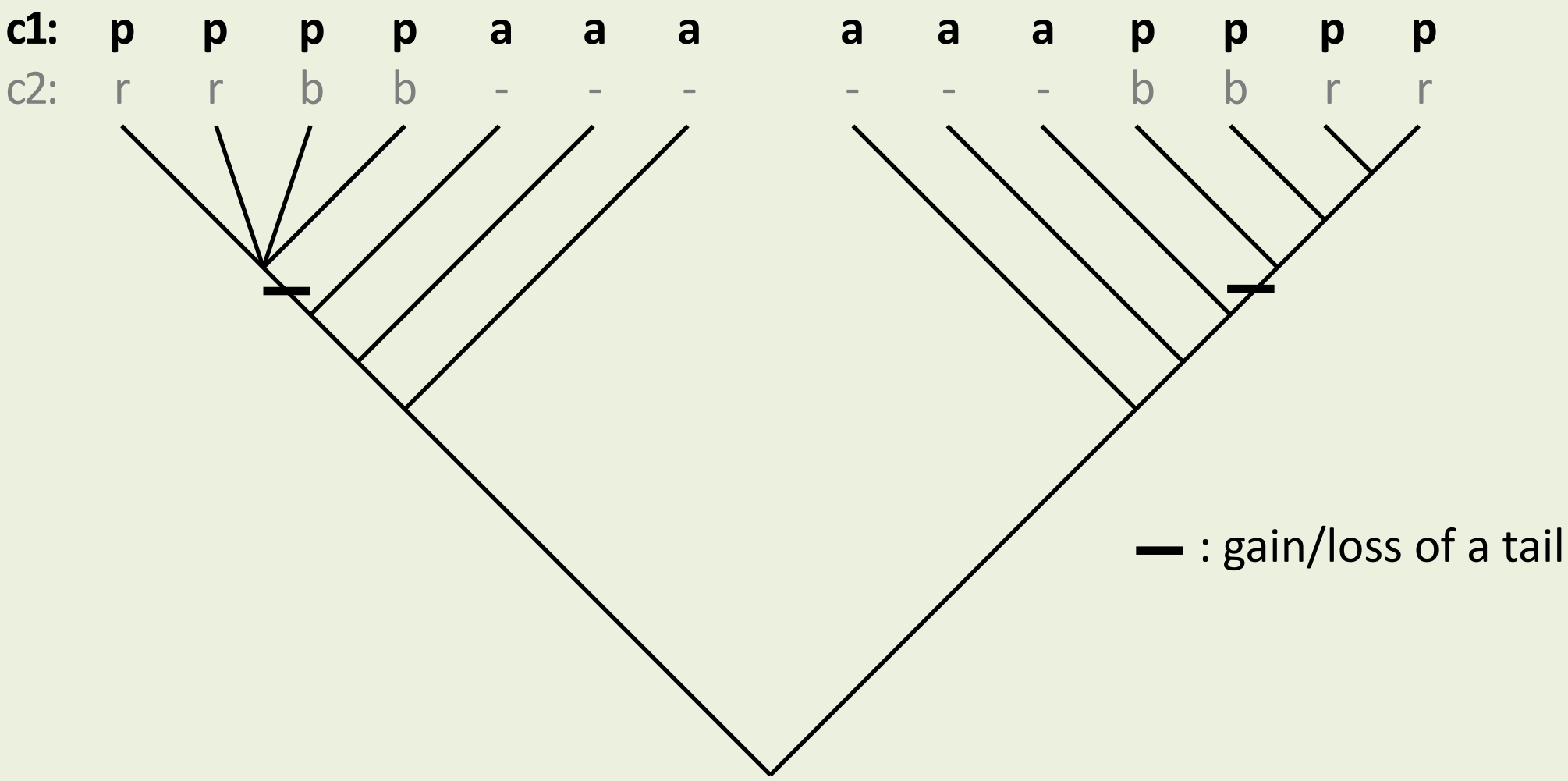
c1: tail (a) absent (p) present

c2: tail color: (-) inapplicable (r) red (b) blue



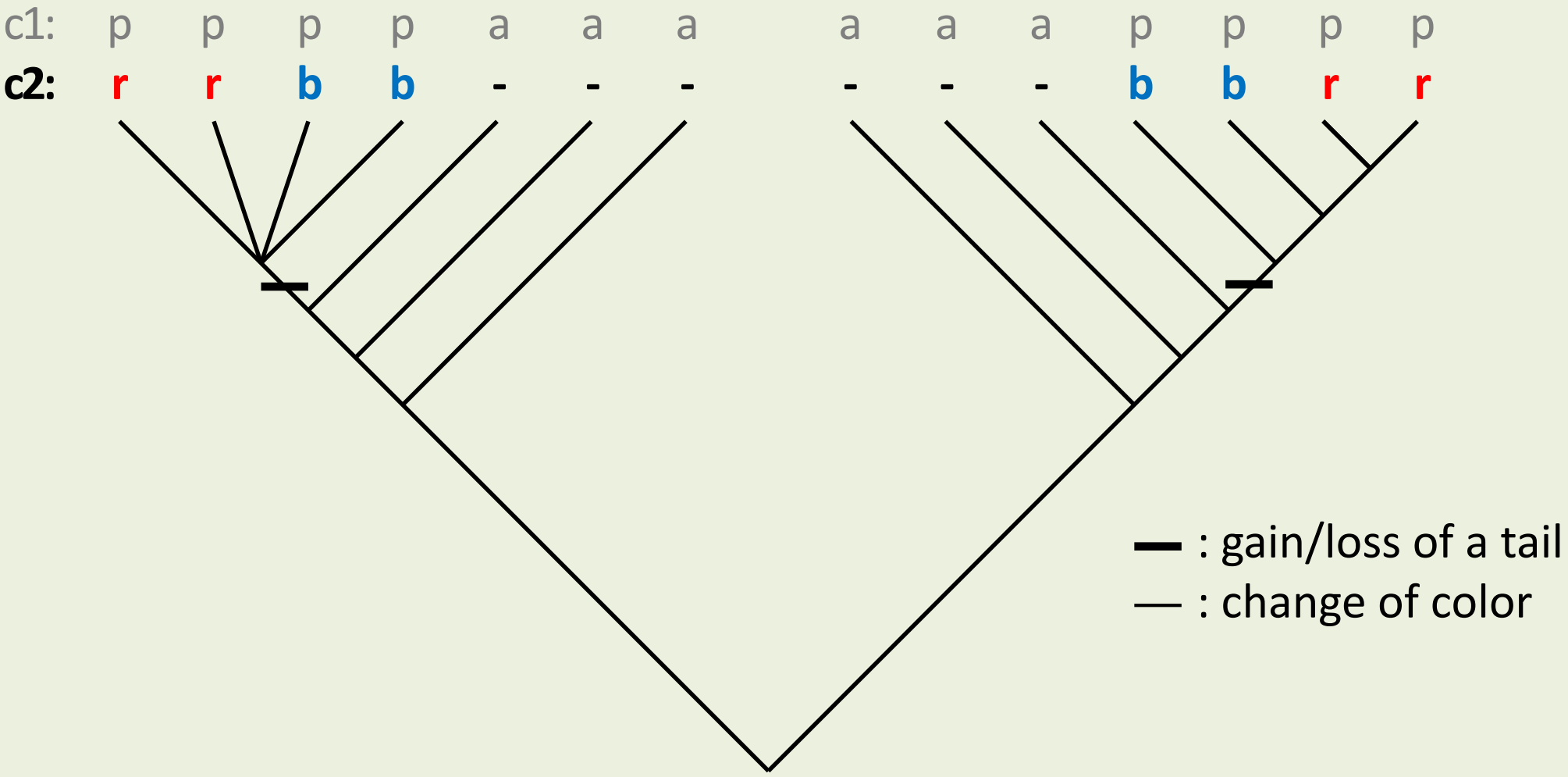
c1: tail (a) absent (p) present

c2: tail color: (-) inapplicable (r) red (b) blue



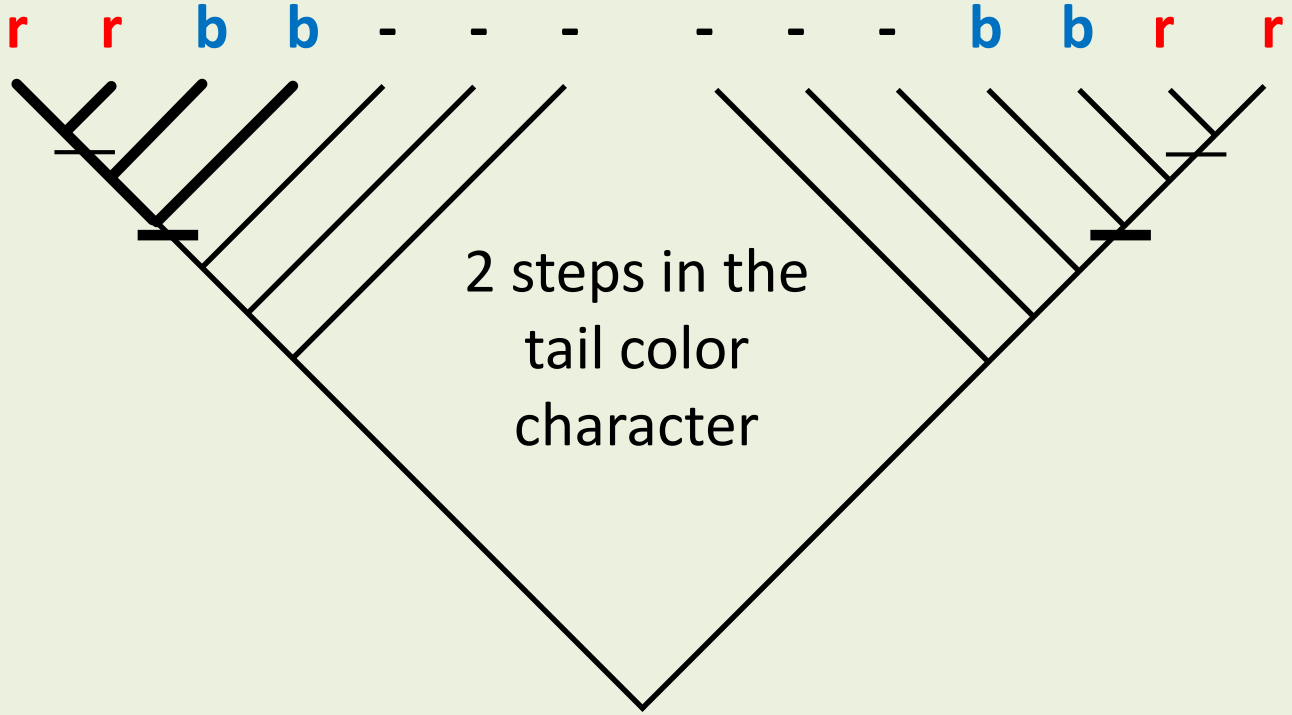
c1: tail (a) absent (p) present

c2: tail color: (-) inapplicable (r) red (b) blue



c1: tail (a) absent (p) present

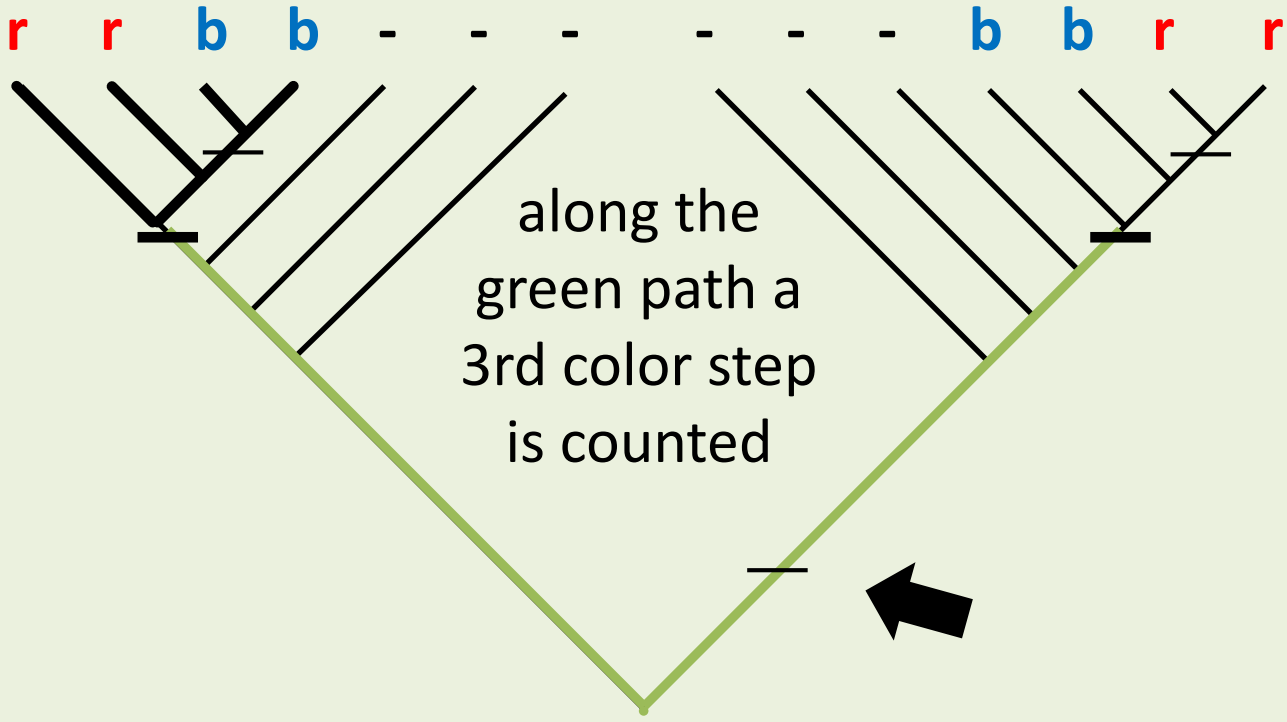
c2: tail color: (-) inapplicable (r) red (b) blue



— : gain/loss of a tail

— : change of color

The **(r (r (b b)))** resolution is rejected because of an unwanted long distance effect of a clade with non-homologous tails.



Other ways of coding such cases into one or more additive or non-additive characters exist, but none of them provides a *general* solution.

See e.g.

- **Maddison. 1993**. Missing data versus missing characters in phylogenetic analysis. *Systematic Biology* 42: 576-581.
- **Strong and Lipscomb. 1999**. Character coding and inapplicable data. *Cladistics* 15: 363-371.
- **Lee and Bryant. 1999**. A reconsideration of the coding of inapplicable characters: assumptions and problems. *Cladistics* 15: 373-378.
- **Sereno. 2007**. Logical basis for morphological characters in phylogenetics. *Cladistics* 23: 565-587.

"Perhaps the eventual solution will be to write new algorithms for computer programs that will allow the characters to be coded independently but that will consider interactions between characters and count steps in some characters only on those portions of the tree on which they are applicable."

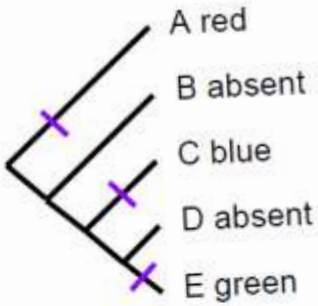
Maddison W.P. 1993. Missing data versus missing characters in phylogenetic analysis. Systematic Biology 42: 576-581.

Parsimony algorithms for characters that are inapplicable in some terminals. Jan De Laet, Department of Invertebrates, American Museum of Natural History, Central Park West at 79th Street, New York, New York 10024-5192, USA.

Abstracts of the **21st Annual Meeting of the Willi Hennig Society (Helsinki, 2002)** *Cladistics* 19:151, 2003.

Minimizing total steps
versus
minimizing extra steps

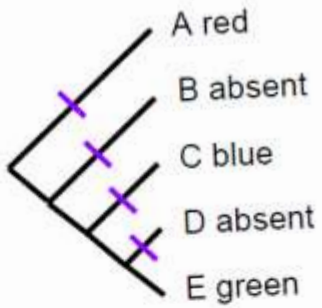
can give different answers in such cases.



3 steps

ad hoc assumptions of homoplasy

tail A/P	2
tail color	0
total	2

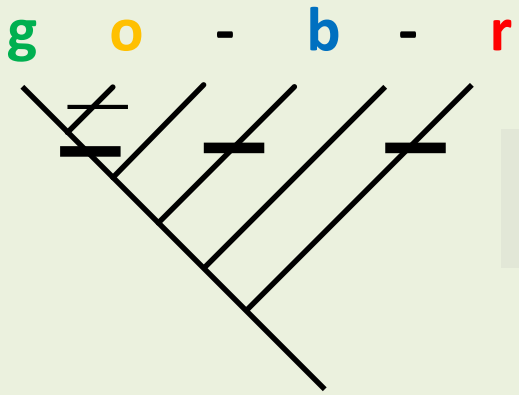


4 steps

ad hoc assumptions of homoplasy

tail A/P	1
tail color	0
total	1

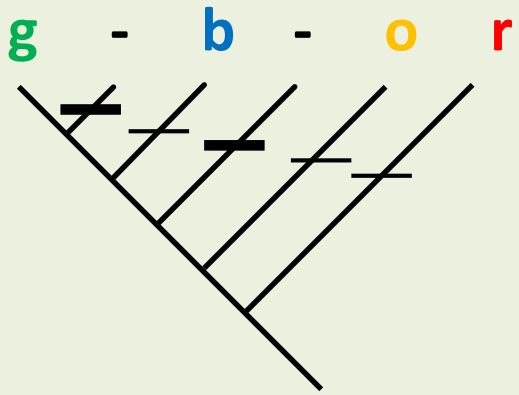
tree 1



— : change of color

▬ : gain/loss of a tail

tree 2

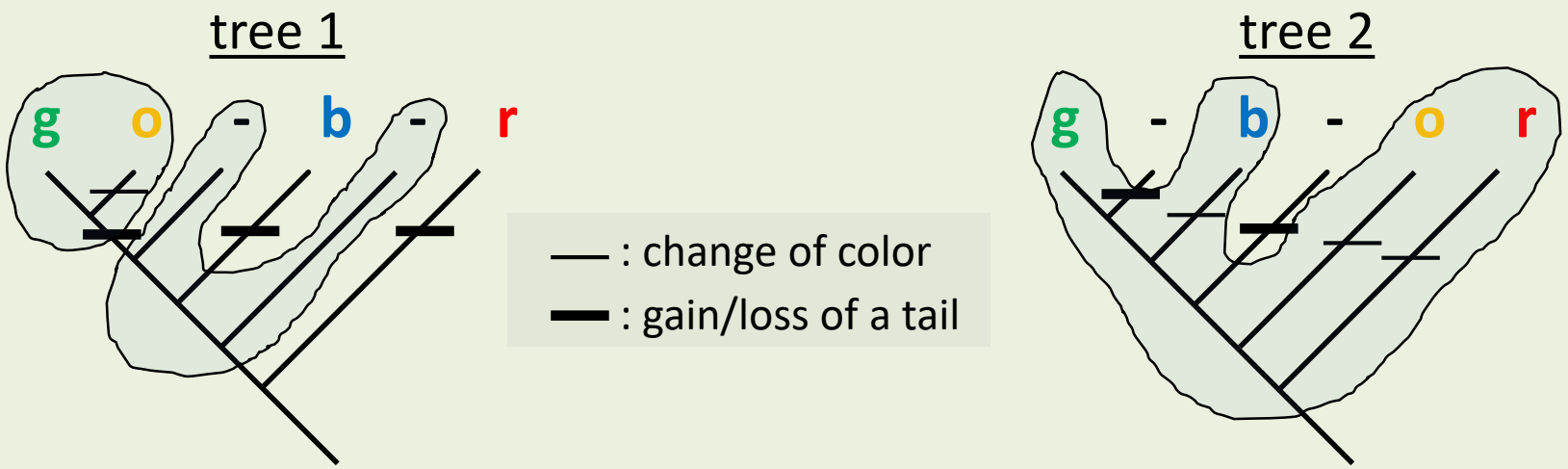


⇨ 4 steps (one optimization shown)

2 extra steps: 3 tail gains (not 1)

5 steps (one optimization shown)

⇨ 1 extra step: 2 tail losses (not 1)



⇨ 4 steps (one optimization shown)

2 extra steps: 3 tail gains (not 1)

2 independent pairwise similarities
are explained by inheritance
and common descent:
shared tail absence in 2 terminals
shared tail presence in 2 terminals

5 steps (one optimization shown)

⇨ 1 extra step: 2 tail losses (not 1)

⇨ 3 independent pairwise similarities
are explained by inheritance
and common descent:
shared tail presence in 4 terminals

It looked obvious that the proper way out of this was

to minimize extra steps

as a means

to maximize homology.

One could at will ‘insert’ complete phenotypes with a single ’indel event’ otherwise – and end up explaining nothing at all ...

I followed up that lead by the next Hennig Meeting (NY 2003).

Tree alignments: analysis of sequence data without prior alignments.

To maximize the equally weighted amount of similarity that can be interpreted as homology, one has to minimize simultaneously the number of

- indel events
- substitutions
- subcharacters

De Laet 2004.
When one and one is not two: parsimony analysis of sequence data.
Abstracts of the **22nd Annual Meeting of the Willi Hennig Society (NY, 2003)**
Cladistics 20: 81.

De Laet 2005.
Parsimony and the problem of inapplicables in sequence data.
Pp. 81-116 in Albert (ed.) Parsimony, Phylogeny, and Genomics. OUP.

Tree alignments: analysis of sequence data without prior alignments.

To maximize the equally weighted amount of similarity that can be interpreted as homology, one has to minimize simultaneously the number of

- indel events
- substitutions
- subcharacters

- in programs such as POY, you get a good enough approximation by using

  substitution cost 2
  gap opening cost 3
  gap extension cost 1

⇨ up to three sequences, these two different criteria give the same numerical result – and POY never aligns more than three sequences at a time

Parsimony is about maximizing homology in your data

- Minimizing extra steps or homoplasy are guaranteed to get you there when you have no inapplicables.

- In the more general case with inapplicables, you'll get there with simultaneous minimization of indels, substitutions, and subcharacters

In both cases the fundamental underlying rationale is maximization of conformity between observational data and explanation.

This point is missed in some recent papers. A good example thereof:

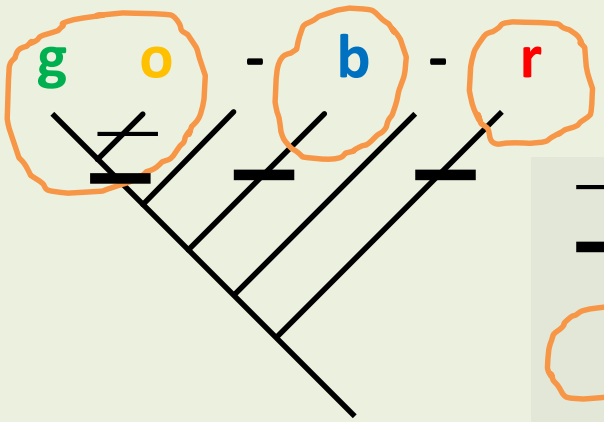
Wheeler, W.C. 2012
Trivial minimization of extra-steps under dynamic homology
Cladistics 28(2): 188-189.

How to do this with inapplicables
as commonly encountered in morphology?

To maximize the amount of similarity that can be interpreted as homology, one has to minimize simultaneously the number of

- indel events ➡ losses and gains of a tail
- substitutions ➡ changes in color where tail color is applicable
- subcharacters ➡ number of distinct regions on the tree in which tail color is applicable.

3 subcharacters for tail color

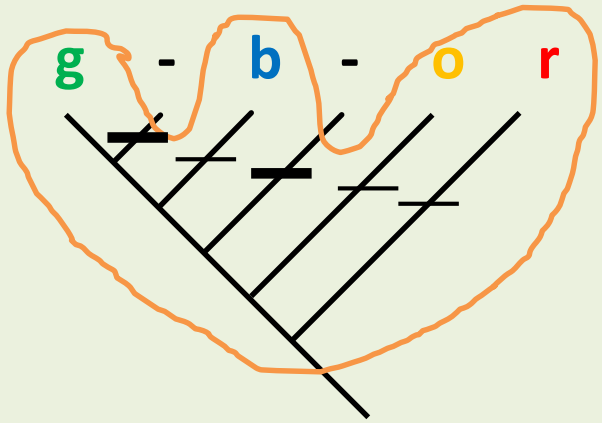


: change of color

: gain/loss of a tail

: a tail-color subcharacter

1 subcharacter for tail color



⇨ 4 steps (one optimization shown)

2 extra steps: 3 tail gains (not 1)

2 independent pairwise similarities are explained by inheritance and common descent:
shared tail absence in 2 terminals
shared tail presence in 2 terminals

tail gains/losses + color changes + subcharacters: 3 + 1 + **3** = 7

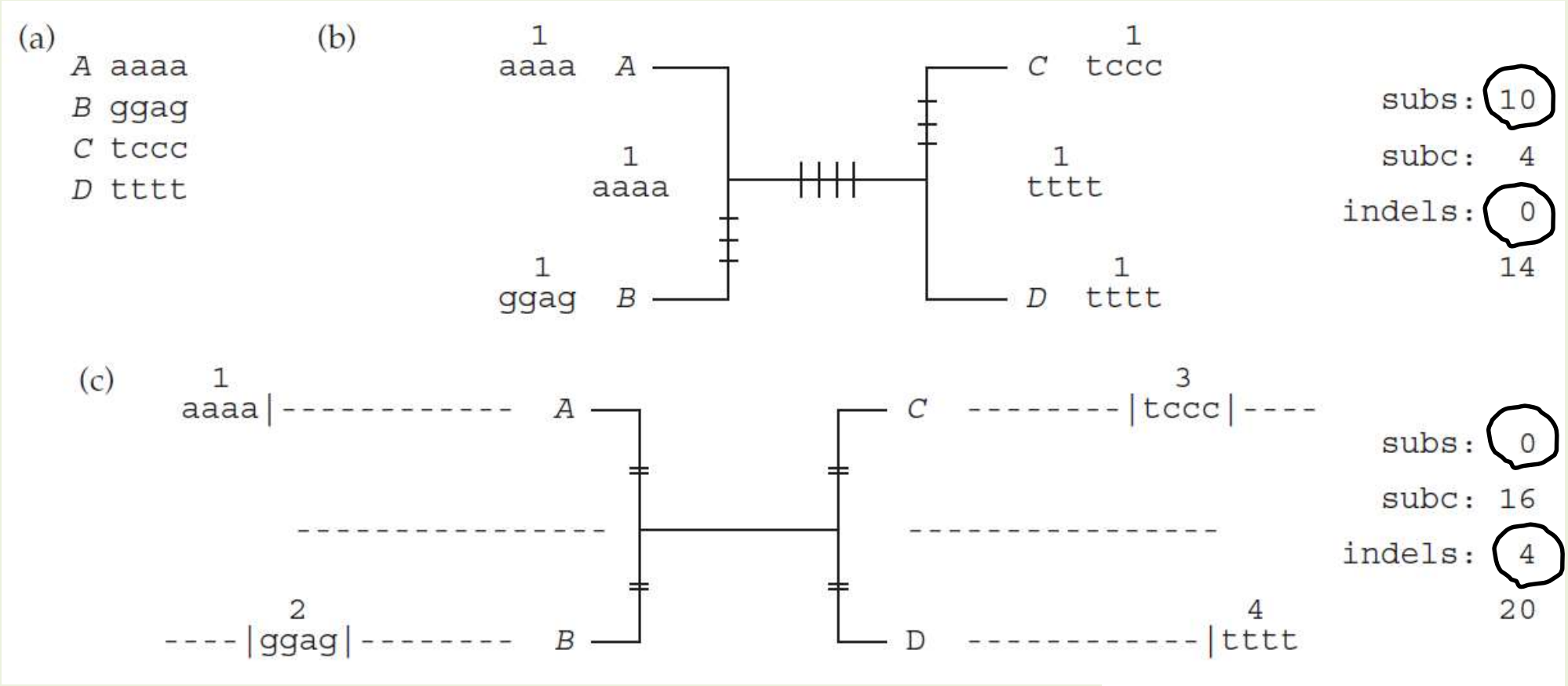
5 steps (one optimization shown)

⇨ 1 extra step: 2 tail losses (not 1)

⇨ 3 independent pairwise similarities are explained by inheritance and common descent:
shared tail presence in 4 terminals

⇨ tail gains/losses + color changes + subcharacters: 2 + 3 + **1** = 6

If instead steps (as evolutionary events) are minimized in such cases, in the end nothing gets explained at all....



(De Laet 2005, fig. 6.13.)

These could equally well be 4 characters that describe 4 aspects of a tail when it is present.

one should be counting

an indel of length n as n evolutionary events

Mostly in various papers of

Kluge & Grant and Grant & Kluge

Besides being a strong and hard to defend knowledge claim about the processes that shape evolution,

it is also easily shown to give less than satisfactory results in practice.

A ttaatt
B ttaaatt
C ttaaaatt
D ttaaaaatt

"[When counting an indel of length n as n steps], unrooted tree (A B)(C D) is preferred because, operationally, it best groups the series of a's in the middle of the observed sequences according to their length. With the cost regime that maximizes homology, the three different unrooted trees for four terminals are considered equally good explanations"

(De Laet 2005: 113)

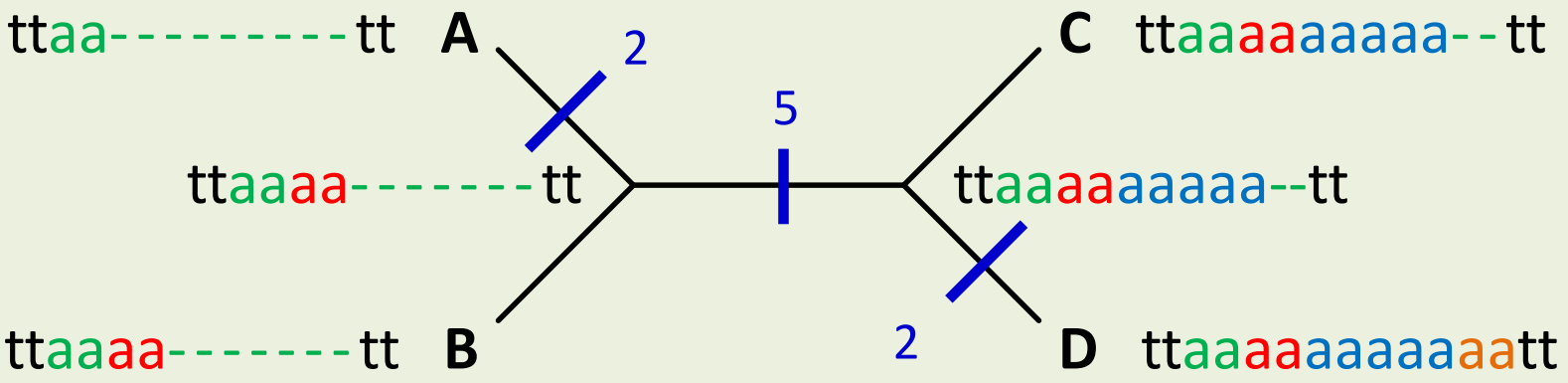
```
A  ttaa----------tt
B  ttaaaa-------tt
C  ttaaaaaaaaa--tt
D  ttaaaaaaaaaaatt
```

This alignment can be shown to be among the many different optimal alignments that exist for this dataset.

Optimal in the sense I've been using throughout: it will yield the tree(s) with maximal homology.

— : indels (of various lengths)

A ttaa---------tt
B ttaaaa-------tt
C ttaaaaaaaaa--tt
D ttaaaaaaaaaaatt

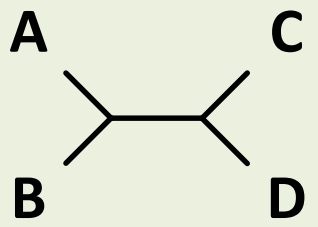


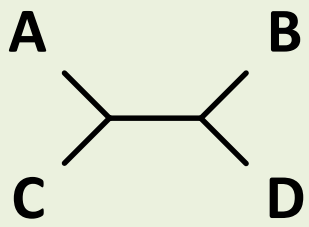
3 indels (other optimizations with 3 indels exist)
0 substitutions
11 subcharacters (1 region of presence for each green position)

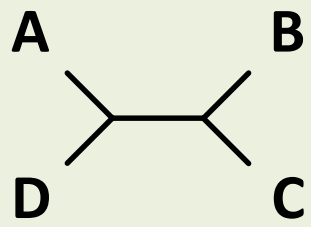
total score **14**

→ this minimization maximizes the possible interpretation of the data in terms of subsequences (and individual bases) that reflect a common evolutionary history

```
A  ttaa----------tt
B  ttaaaa-------tt
C  ttaaaaaaaaa--tt
D  ttaaaaaaaaaaatt
```





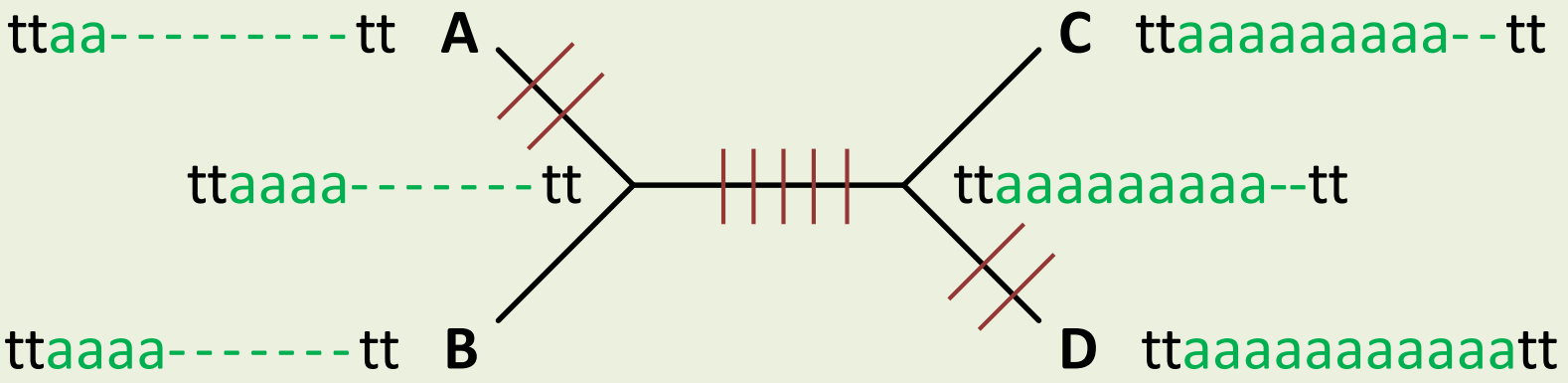


minimize indels + substitutions + subcharacters to maximize homology

14 **14** **14**

— : indels of single positions

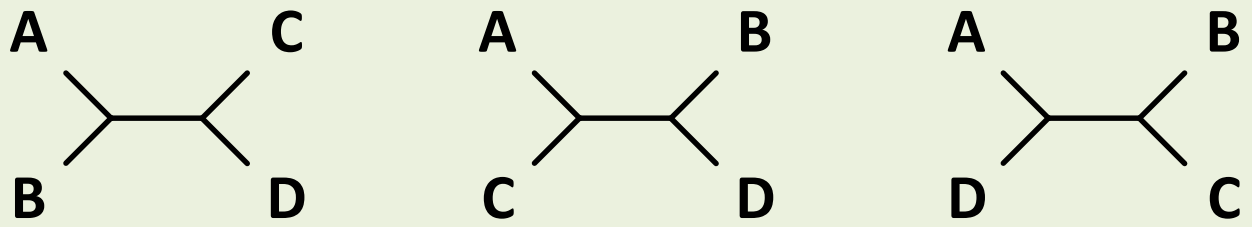
A ttaa---------tt
B ttaaaa-------tt
C ttaaaaaaaaa--tt
D ttaaaaaaaaaaatt



9 ‘unit’ indel events (other such optimizations exist)

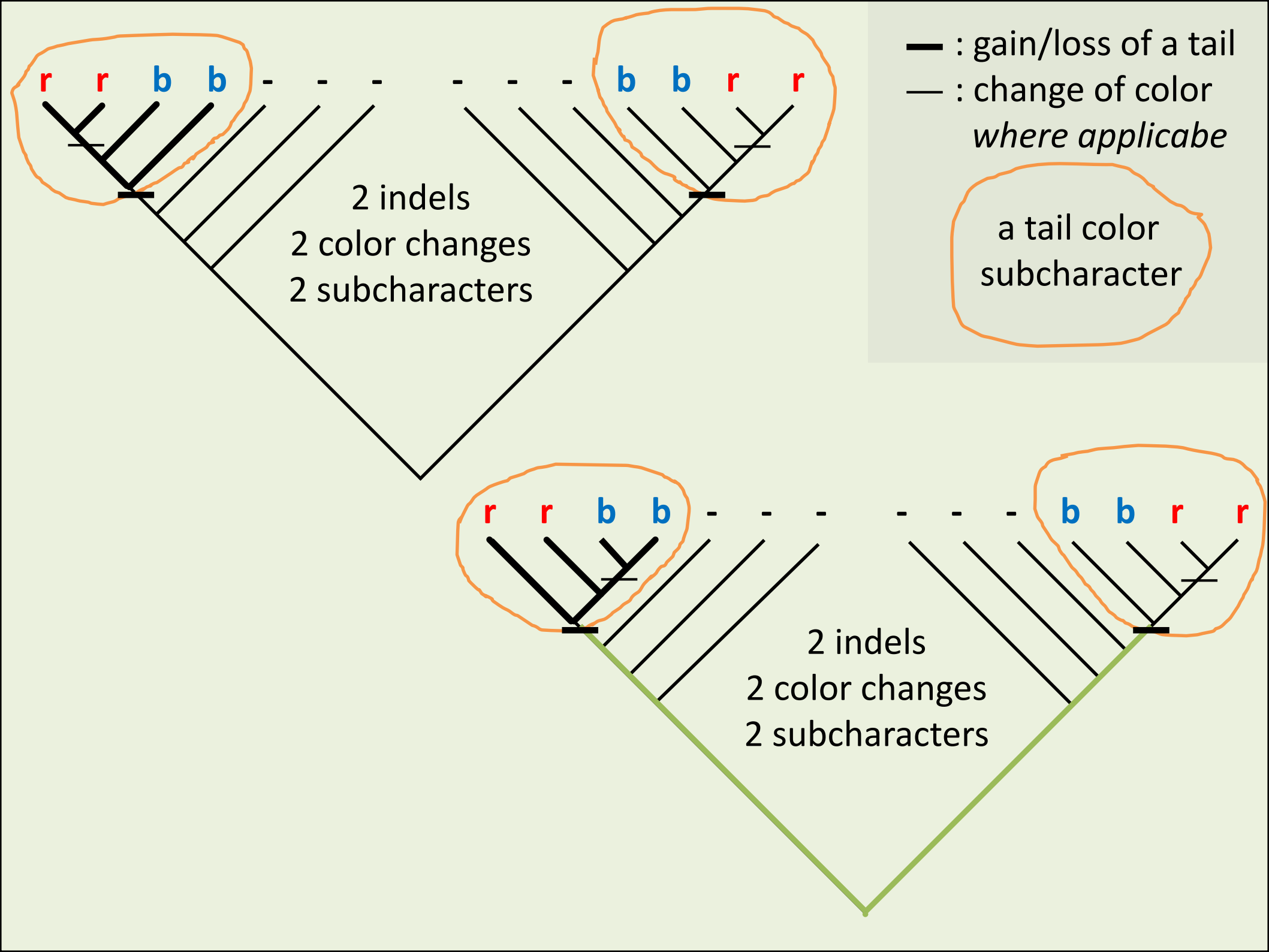
→ this amounts to minimizing as, an additive character, the length of the subsequences that take part in indel events

```
A  ttaa----------tt
B  ttaaaa-------tt
C  ttaaaaaaaaa--tt
D  ttaaaaaaaaaaatt
```



minimize indels + substitutions + subcharacters to maximize homology	**14**	**14**	**14**
minimize 'unit' evolutionary events	**9**	14	16

→ the effect also occurs when using the *APE* parameter set in in POY (***A***ll ***P***arameters ***E***qual)

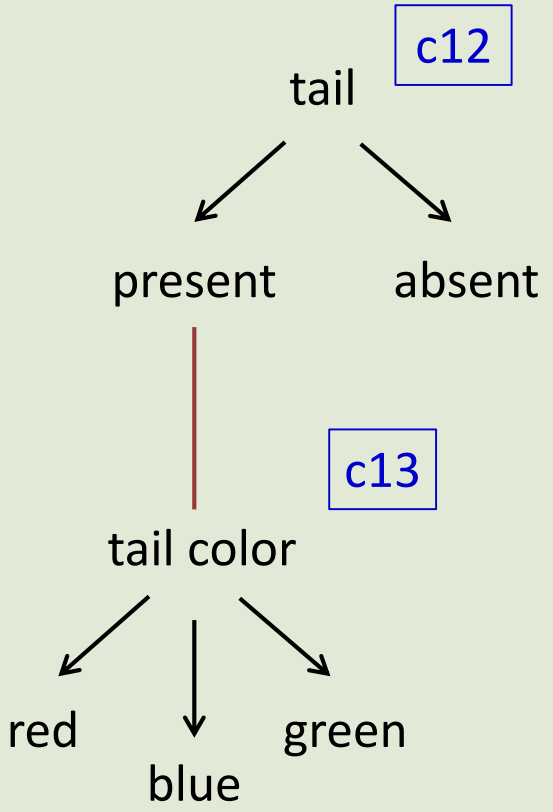

r r b b - - - - - - b b r r
2 indels
2 color changes
2 subcharacters
— : gain/loss of a tail
— : change of color
where applicabe
a tail color
subcharacter
r r b b - - - - - - b b r r
2 indels
2 color changes
2 subcharacters

These ideas are implemented in
anagallis, a computer program for parsimony analysis

- specification of character hierarchies
- tree evaluation and tree search with character hierarchies
- plotting final states of characters in a hierarchy

Specifying a character hierarchy

```
characters input numeric
13 15
out 00000000000 00
A   11111000000 11
B   11111000000 11
C   11111000000 12
D   11111000000 12
E   11110000000 00
F   11100000000 00
G   11000000000 00
H   10000100000 00
I   10000110000 00
J   10000111000 00
K   10000111100 12
L   10000111110 12
M   10000111111 11
N   10000111111 11
;
characters code <12 13>;
```

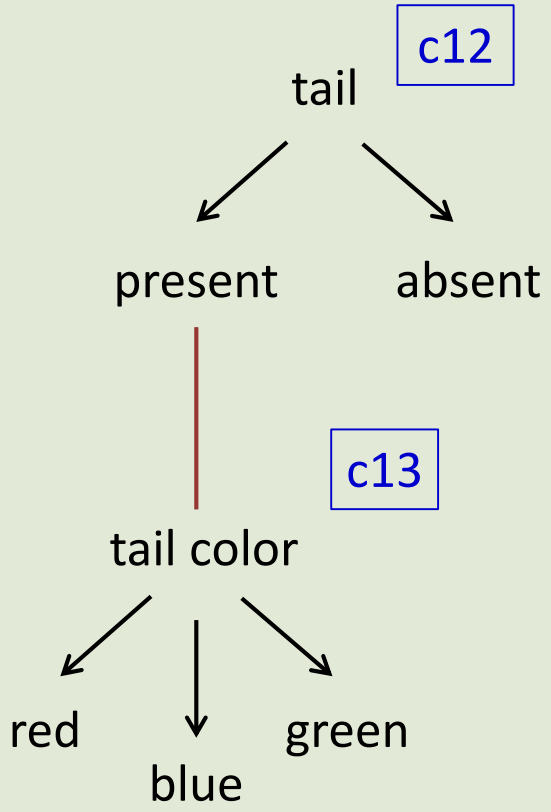


Specifying a character hierarchy

```
>-< cc <12 13>;
```

Part of the response:

```
characters code> 1 character hierarchy
with a total of 2 characters
characters code
  <12:0 13:0>;
```

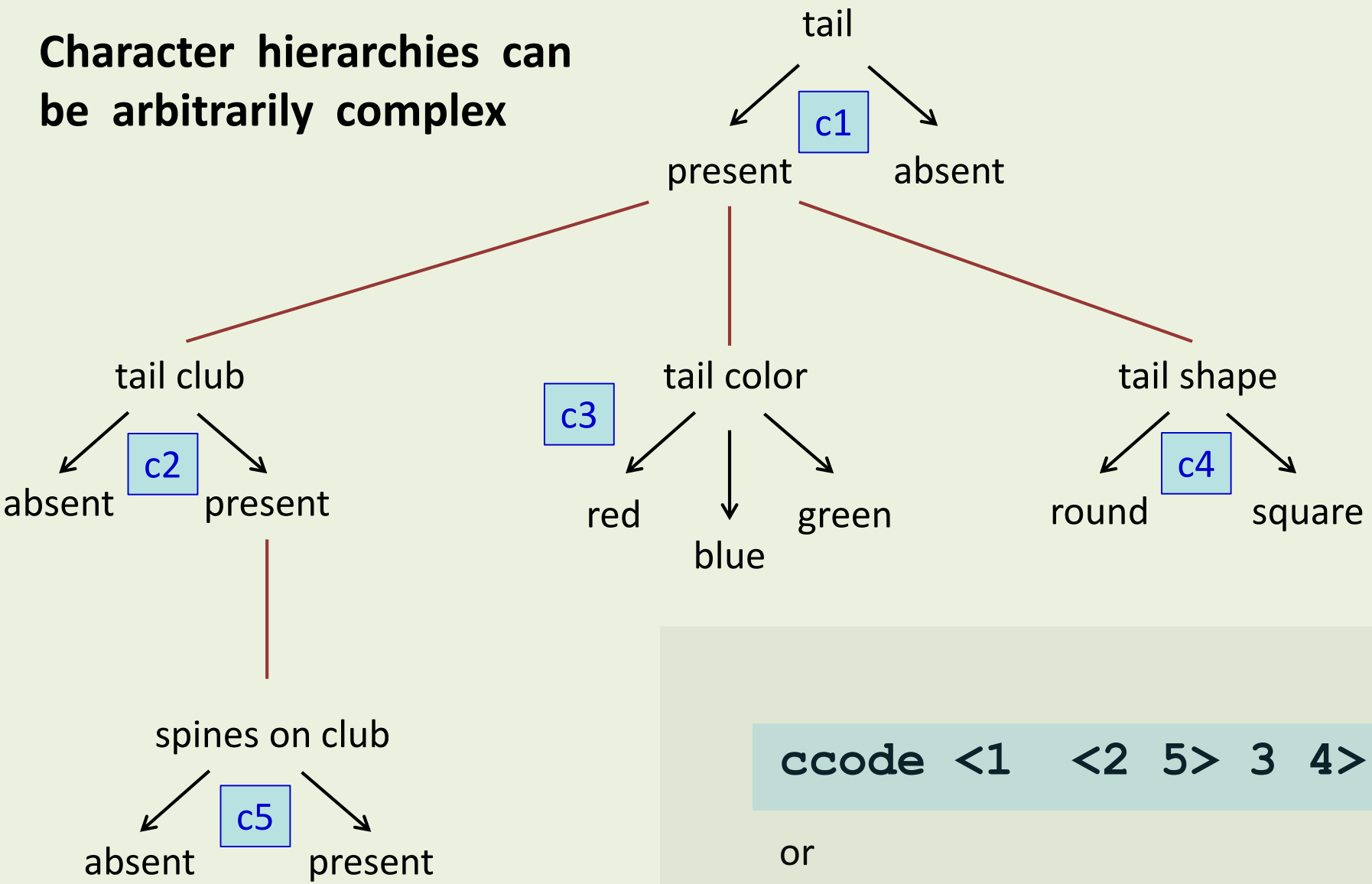


The program now knows that

- character 12 is an absence/presence character
- character 13 is subordinate to character 12
- the character code to indicate absence or inapplicability is ‘0’ (this is a default that can be overruled, character by character).

It will flag an error if the state distributions of c12 and c13 contradict this interpretation

Character hierarchies can be arbitrarily complex

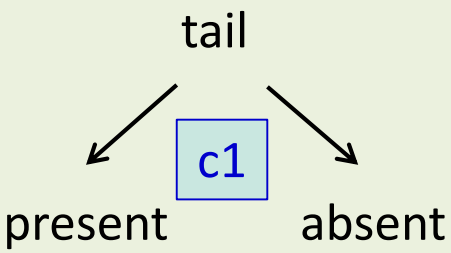


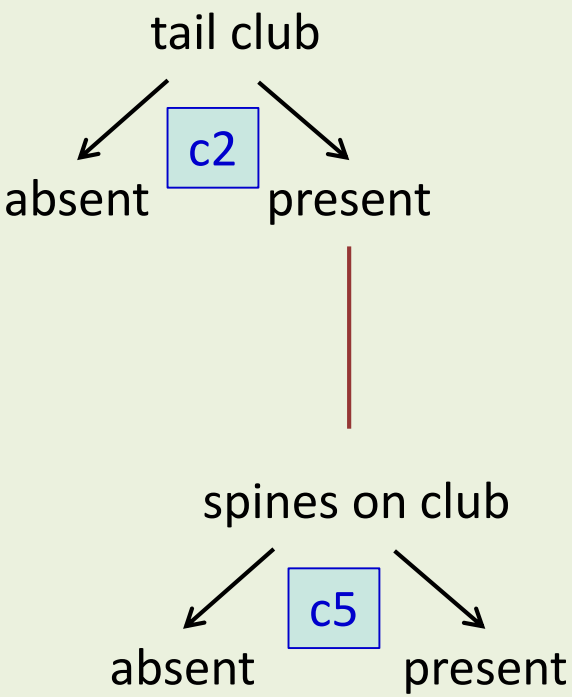

```
ccode <1  <2 5> 3 4>
```

or

```
ccode <1  <2 <5>> 3 4>
```

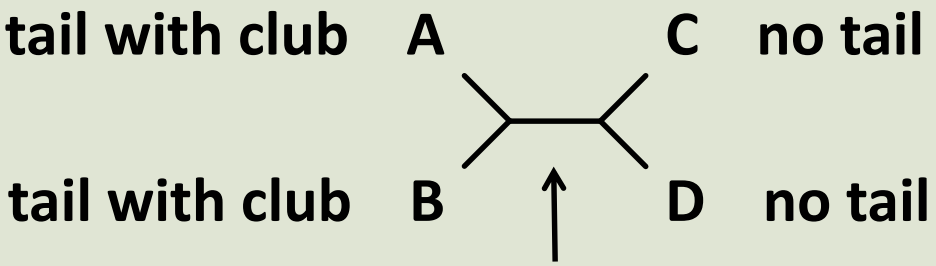
Character hierarchies can be arbitrarily complex





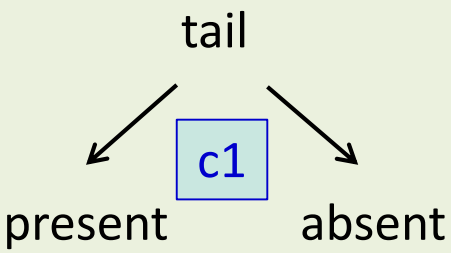
`ccode <1 <2 <5>>>`

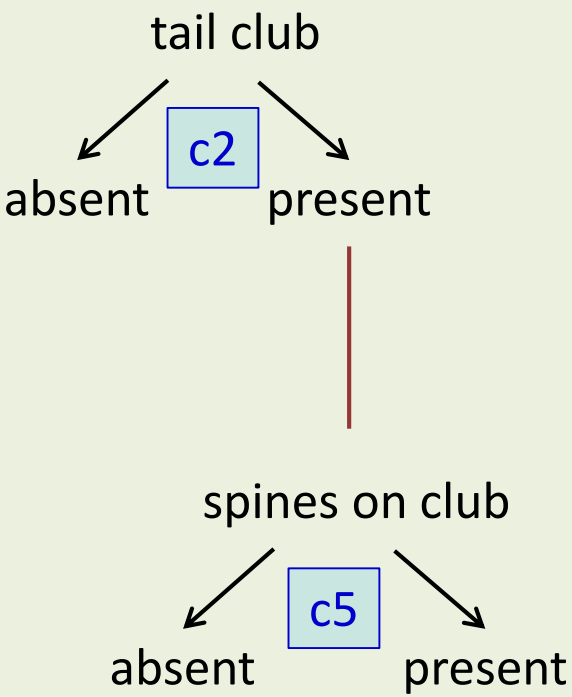
Moving up/down an absence/presence hierarchy along a branch in a tree



1 indel by default

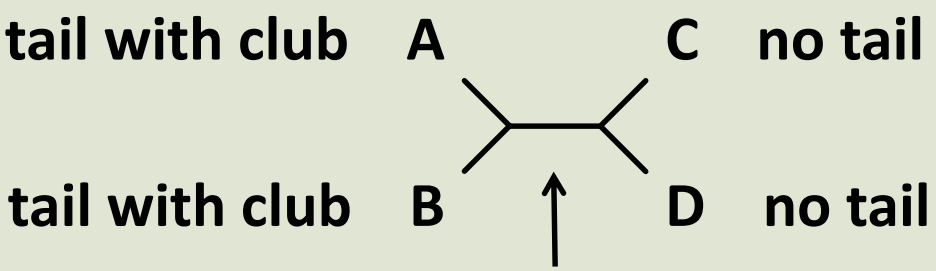
Character hierarchies can be arbitrarily complex






```
ccode <1  <2 <5>>>
```

Moving up/down an absence/presence hierarchy along a branch in a tree



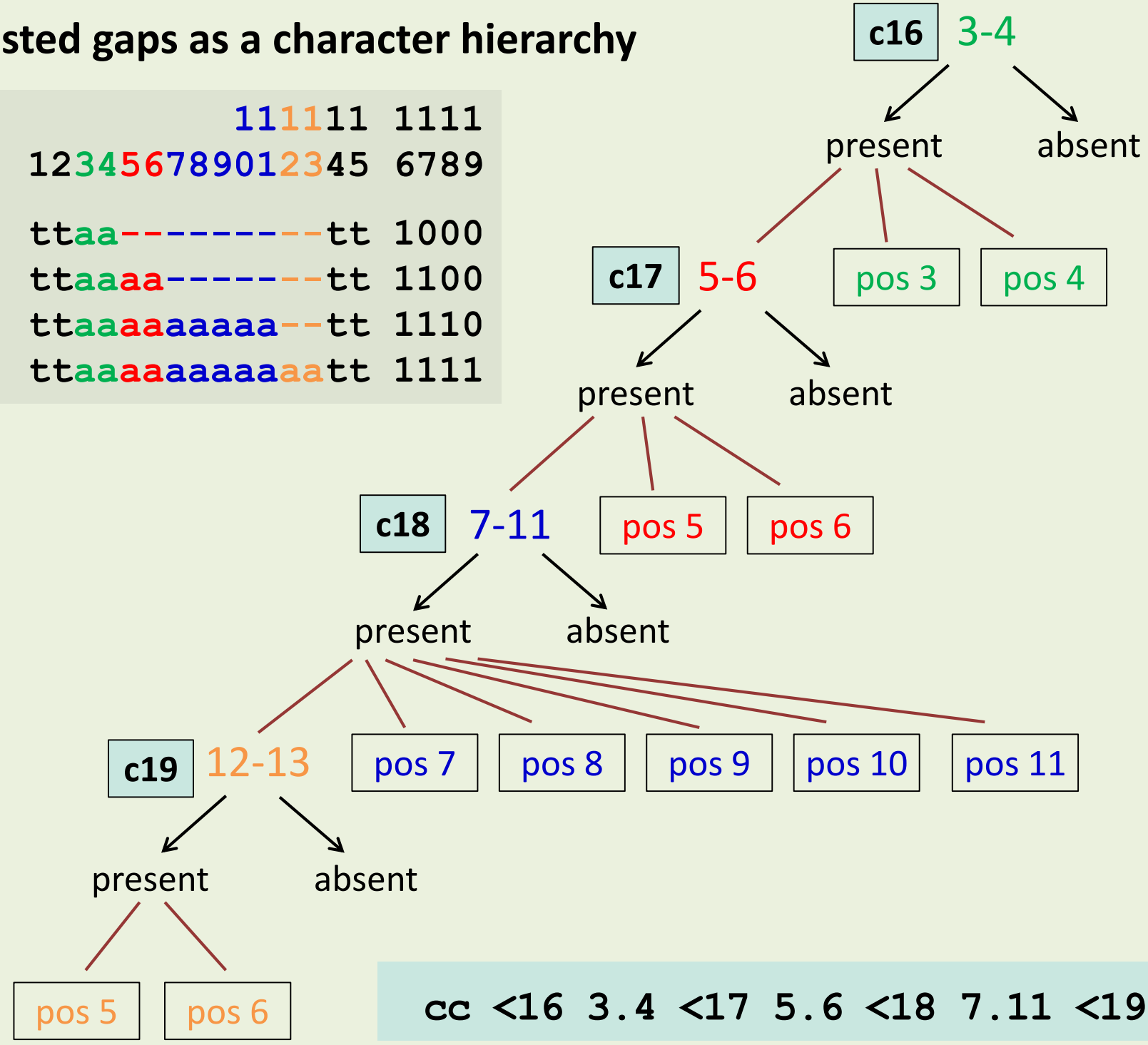
1 indel by default

```
ccode + 2;
```

⟶ count this as 2 indels

Nested gaps as a character hierarchy

```
                111111 1111
    12345678901234 5 6789

A   ttaa--------tt 1000
B   ttaaaa------tt 1100
C   ttaaaaaaaa--tt 1110
D   ttaaaaaaaaaatt 1111
```




```
cc <16 3.4 <17 5.6 <18 7.11 <19 12.13>>>>;
```

These ideas are implemented in
anagallis, a computer program for parsimony analysis

- specification of character hierarchies
- tree evaluation and tree search with character hierarchies
  - a mechanism to ensure that the overall explanation is free of contradictions: constrained downpass and uppass
  - a mechanism to ensure that the overall explanation is optimal
- plotting final states of characters in a hierarchy

Getting the score of a character hierarchy on a tree

1. The number of gains/losses: regular downpass of the A/P-character

2. Determine the number of regions of presence:
   - regular uppass of the A/P character (constrained when nested)
   - resolve any ambiguities as absence (more about that right away)

3. For each subordinate character
   - the number of subcharacters = the number of regions of presence
   - the number of steps in the subcharacters follows from a constrained downpass:

     when visting a node (postorder),
     - if the node is in a region of absence,
       the preliminary state is 'inapplicabble'
     - else if either daughter is in a region of absence,
       pass through the preliminary stateset of the other daughter
     - else
       proceed with a regular downpass.

Determining final state sets for regular subordinate characters

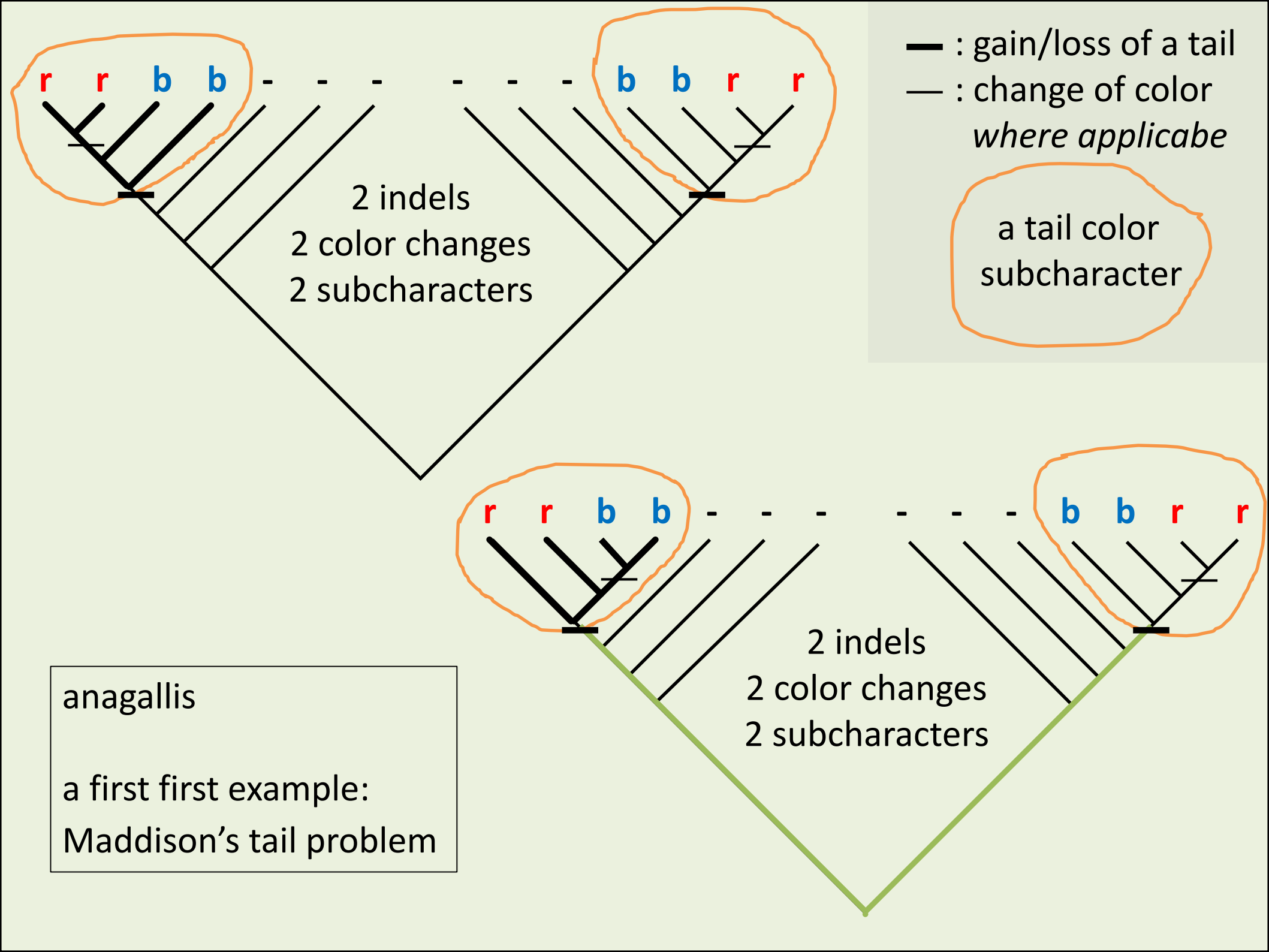
- initialize the root: the final state set is the preliminary state set
- when visting a node (preorder),
  - if the node is in a region of absence,
    the final state set is ‘inapplicabble’
  - else if the parent has ‘inapplicable’ and this node hasn’t,
    initialize a new region of presence: the final state set is the preliminary state set
  - else
    proceed with a regular uppass

This procedure is guaranteed to return the optimal score when all pairs of regions of presence are separated by at least as many absence nodes as there are subcharacters.

When shorter paths of absence between two regions of presence exist, exist, the score *may* be improved by *flipping* such paths or combinations thereof from absence to presence.

The theory is straightforward, but the flip checks are not yet in the program.

Make sure the explanation is optimal

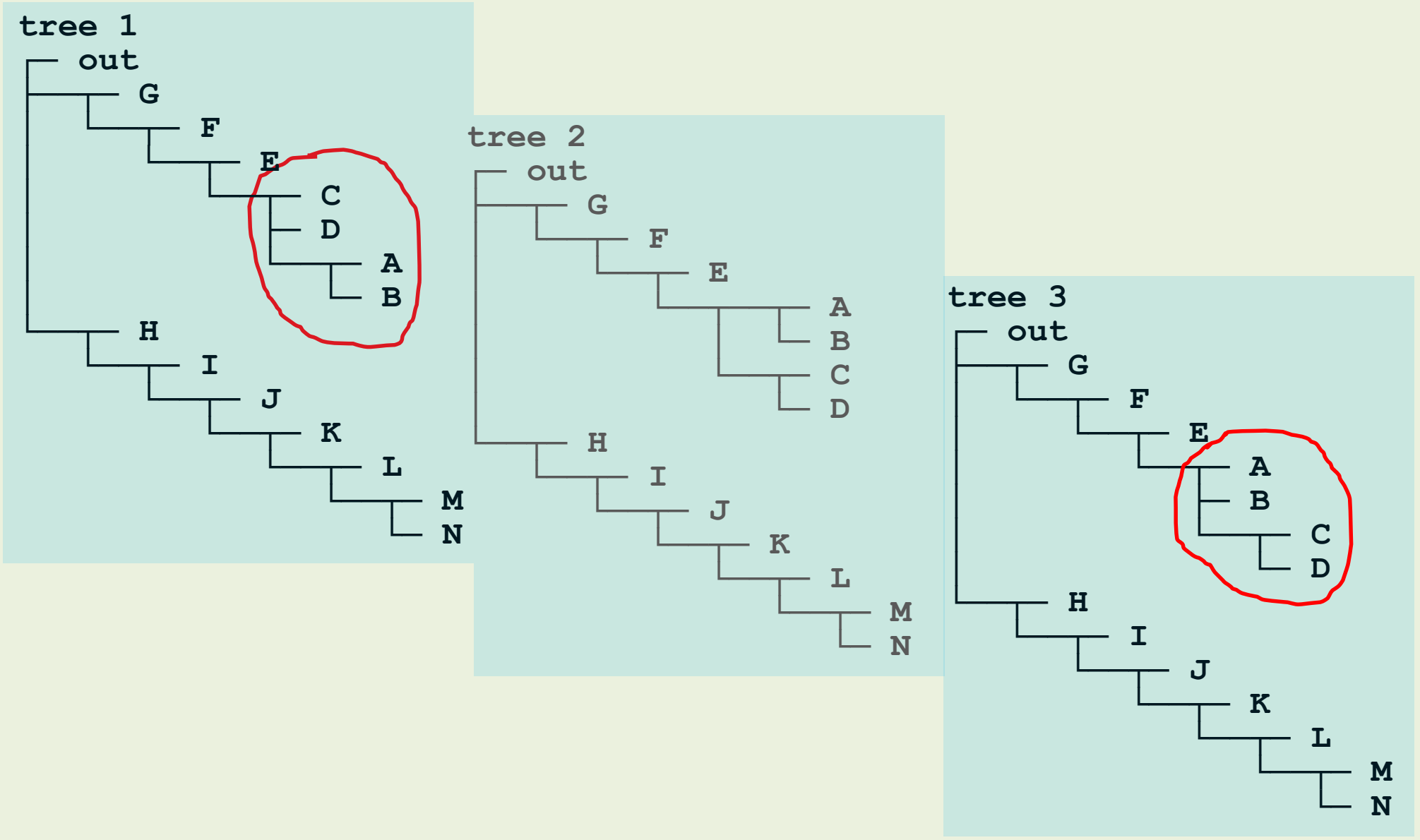

r r b b - - - - - - b b r r
2 indels
2 color changes
2 subcharacters
— : gain/loss of a tail
— : change of color
where applicabe
a tail color
subcharacter
r r b b - - - - - - b b r r
2 indels
2 color changes
2 subcharacters
anagallis
a first first example:
Maddison's tail problem

```
characters input numeric
13 15
out 00000000000 00
A   11111000000 11
B   11111000000 11
C   11111000000 12
D   11111000000 12
E   11110000000 00
F   11100000000 00
G   11000000000 00
H   10000100000 00
I   10000110000 00
J   10000111000 00
K   10000111100 12
L   10000111110 12
M   10000111111 11
N   10000111111 11
;
```

```
characters code <12 13>;

trees set zerocollapse 1;

trees find mult s5

trees show plot ab.

characters diagnose scores .

char diag optimizations plot .abc12

char diag opt plot . ab c13
```

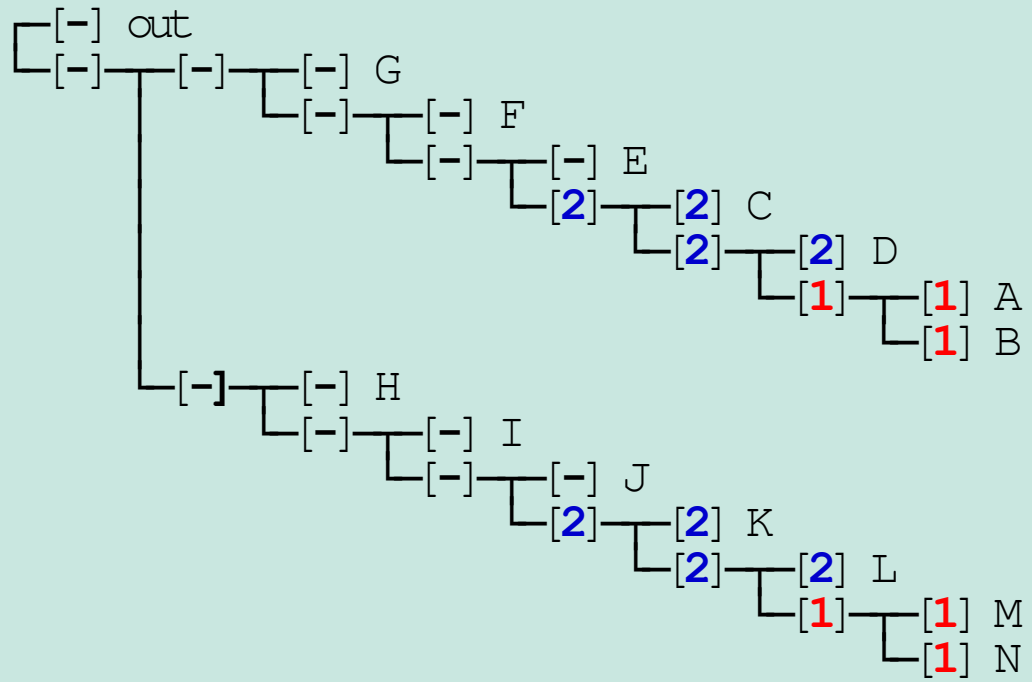
character 12: tail
- absent(0)
- present(1)

character 13: tail color
- inapplicable (0)
- red(1)
- blue (2)

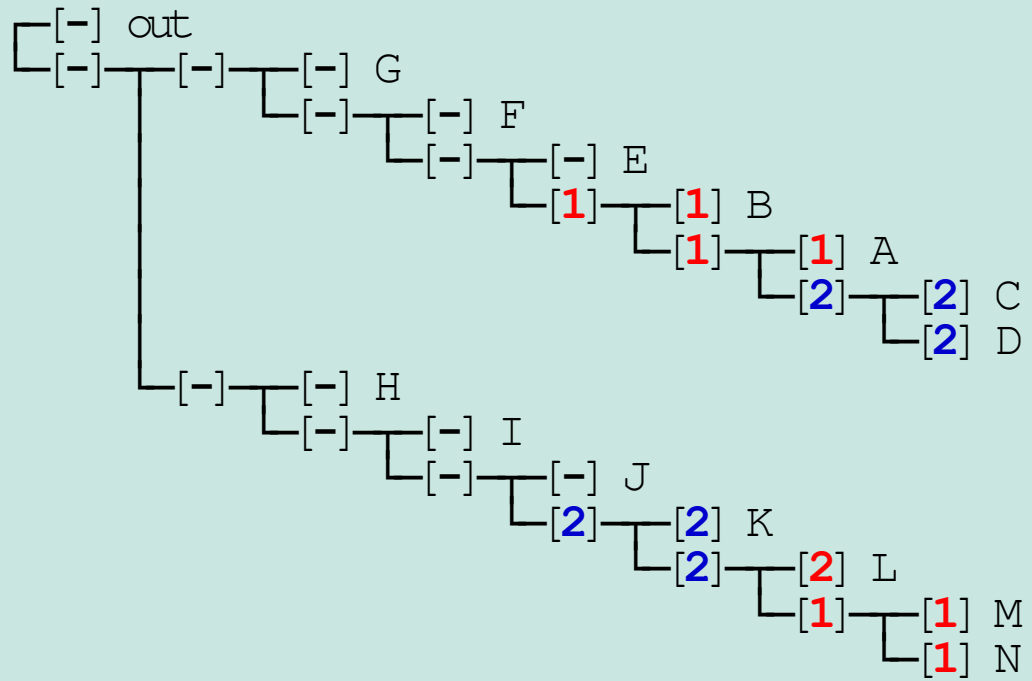

tree 1
out
G
F
E
C
D
A
B
H
I
J
K
L
M
N
tree 2
out
G
F
E
A
B
C
D
H
I
J
K
L
M
N
tree 3
out
G
F
E
A
B
C
D
H
I
J
K
L
M
N

```
characters diagnose scores> diagnosing tree scores for trees 1.3
characters diagnose scores> tree 1
0 1 1 1 1 1 1 1 1 1 1 2 4
total score: 16
scores of character hierarchies:
  * <12:0 13:0>
  * series at 12: total score 6 (2 indels, 2 subsitutions, 2 subcharacters)
characters diagnose scores> tree 2
0 1 1 1 1 1 1 1 1 1 1 2 4
total score: 16
scores of character hierarchies:
  * <12:0 13:0>
  * series at 12: total score 6 (2 indels, 2 subsitutions, 2 subcharacters)
characters diagnose scores> tree 3
0 1 1 1 1 1 1 1 1 1 1 2 4
total score: 16
scores of character hierarchies:
  * <12:0 13:0>
  * series at 12: total score 6 (2 indels, 2 subsitutions, 2 subcharacters)
```

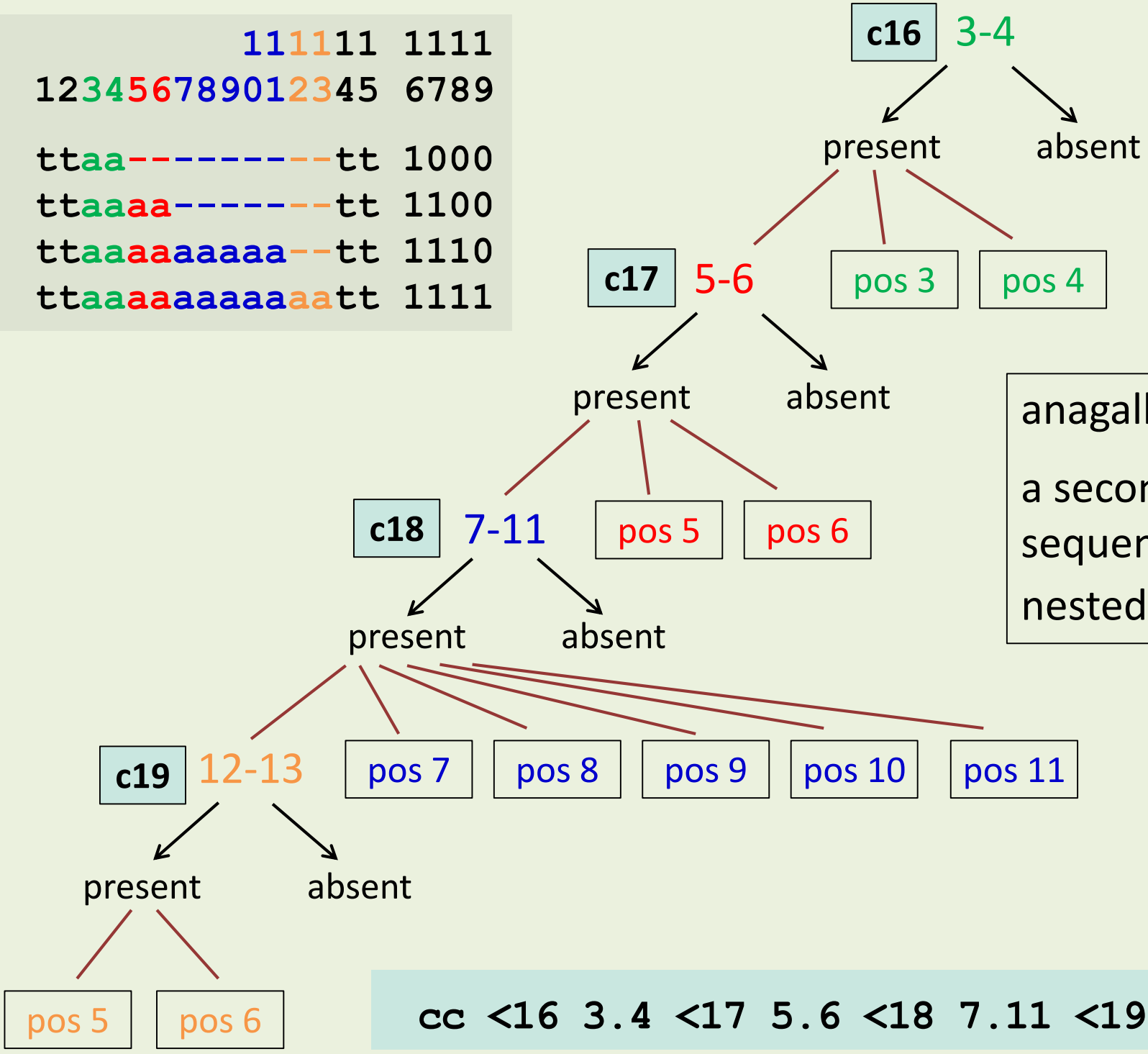
character 13 on tree 1:



character 13 on tree 3:



```
                 111111 1111
    12345678901234 5 6789

A   ttaa---------tt 1000
B   ttaaaa-------tt 1100
C   ttaaaaaaaaa--tt 1110
D   ttaaaaaaaaaaatt 1111
```

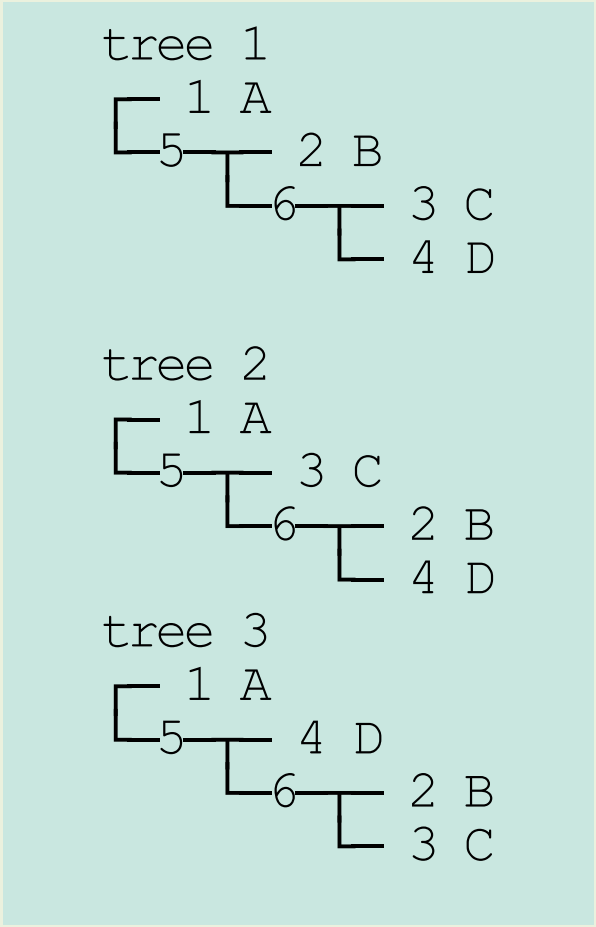


anagallis

a second example: sequences with nested gaps

```
cc <16 3.4 <17 5.6 <18 7.11 <19 12.13>>>>;
```

```
characters input numeric
'This dataset and its accompanying ccode command describe
sequences
   ttaatt,
   ttaaaatt,
   ttaaaaaaaaatt, and
   ttaaaaaaaaaaatt (after de Laet 2005: 113).
Its comparative structure is such that the coded alignment can
be shown to be among the optimal alignments.

Up to character 15, 1 means a, 2 t, and 0 a unit gap.
Characters 16.19 describe the unordered nested set of
subsequences at 3.13
'

19 4
A  221100000000022  0001
B  221111000000022  0011
C  221111111110022  0111
D  221111111111122  1111
;
cc <19 3.4 <18 5.6 <17 7.11 <16 12.13>>>>;
```


tree 1
1 A
5
2 B
6
3 C
4 D
tree 2
1 A
5
3 C
6
2 B
4 D
tree 3
1 A
5
4 D
6
2 B
3 C

```
characters diagnose scores> tree 1
0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0
total score: 14
scores of character hierarchies:
  * <19:0 3:0 4:0 <18:0 5:0 6:0 <17:0 7:0 8:0 9:0 10:0 11:0 <16:0 12:0 13:0>>>>
  * series at 16: total score 3 (1 indels, 0 subsitutions, 2 subcharacters)
  * series at 17: total score 9 (1 indels, 0 subsitutions, 5 subcharacters,
                                 subscore 3 from subseries)
  * series at 18: total score 12 (1 indels, 0 subsitutions, 2 subcharacters,
                                  subscore 9 from subseries)
  * series at 19: total score 14 (0 indels, 0 subsitutions, 2 subcharacters,
                                  subscore 12 from subseries)
```

```
characters diagnose scores> tree 2
0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0
total score: 14
scores of character hierarchies:
  * <19:0 3:0 4:0 <18:0 5:0 6:0 <17:0 7:0 8:0 9:0 10:0 11:0 <16:0 12:0 13:0>>>>
  * series at 16: total score 3 (1 indels, 0 subsitutions, 2 subcharacters)
  * series at 17: total score 9 (1 indels, 0 subsitutions, 5 subcharacters,
subscore 3 from subseries)
  * series at 18: total score 12 (1 indels, 0 subsitutions, 2 subcharacters,
subscore 9 from subseries)
  * series at 19: total score 14 (0 indels, 0 subsitutions, 2 subcharacters,
subscore 12 from subseries)
characters diagnose scores> tree 3
0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0
total score: 14
scores of character hierarchies:
  * <19:0 3:0 4:0 <18:0 5:0 6:0 <17:0 7:0 8:0 9:0 10:0 11:0 <16:0 12:0 13:0>>>>
  * series at 16: total score 3 (1 indels, 0 subsitutions, 2 subcharacters)
  * series at 17: total score 9 (1 indels, 0 subsitutions, 5 subcharacters,
subscore 3 from subseries)
  * series at 18: total score 12 (1 indels, 0 subsitutions, 2 subcharacters,
subscore 9 from subseries)
  * series at 19: total score 14 (0 indels, 0 subsitutions, 2 subcharacters,
subscore 12 from subseries)
```

plans for the following months

- fixing some known bugs
- write code for flips
- finish documentation and manuscript
- improve search strategies
- extension beyond absence/presence characters
- …

- the program should be available at www.anagallis.be by the end of the year