Table 1: An approximation of the probability of data patterns on the tree shown in figure **??** made by dropping terms that do not have the minimal exponent for p. Terms that were dropped are shown in red; Table **??** shows the full (non approximate) probabilities. The final column provides an even rougher approximation by setting $1 - p \approx 1$.

	Internal state (E,F)					
d_i	(0,0)	(0,1)	(1,0)	(1,1)	$\lim_{p\to 0} \Pr(d_i \mid T_{AB})$	$\approx \lim_{p\to 0} \Pr(d_i \mid T_{AB})$
0000	$(1-p)^5$	$(1-p)^2p^3$	$(1-p)^2p^3$	$(1-p)p^4$	$(1-p)^5$	1
0001	$(1-p)^4p$	$(1-p)^3p^2$	$(1-p)p^4$	$(1-p)^2p^3$	$(1-p)^4p$	p
0010	$(1-p)^4p$	$(1-p)^3p^2$	$(1-p)p^4$	$(1-p)^2p^3$	$(1-p)^4p$	p
0011	$(1-p)^3p^2$	$(1-p)^4p$	p^5	$(1-p)^3p^2$	$(1-p)^4p$	p
0100	$(1-p)^4p$	$(1-p)p^4$	$(1-p)^3p^2$	$(1-p)^2p^3$	$(1-p)^4p$	p
0101	$(1-p)^3p^2$	$(1-p)^2p^3$	$(1-p)^2p^3$	$(1-p)^3p^2$	$2(1-p)^3p^2$	$2p^2$
0110	$(1-p)^3p^2$	$(1-p)^2p^3$	$(1-p)^2p^3$	$(1-p)^3p^2$	$2(1-p)^3p^2$	$2p^2$
0111	$(1-p)^2p^3$	$(1-p)^3p^2$	$(1-p)p^4$	$(1-p)^4p$	$(1-p)^4p$	p

Is the equal branch length model a parsimony model?

As we just discussed, when the probability of change is low, the likelihood under our equal-branch length model is dominated by the number of changes. Because we take the product over all patterns to get a dataset's likelihood, the likelihood will be dominated by the sum of the number of steps required to explain the data. This seems to lead us to the conclusion that the parsimony criterion: prefer the tree with the fewest number of steps needed to explain the data is an ML estimator under the equal branch length model.

This is not true – but for almost all datasets that you would encounter it is quite likely that the most parsimonious tree will maximize the equal branch length model. A simple counterexample showing that the MP and ML-equal-branch lengths are not the same is given in table 2. In this example there is a synapormorphy supporting each tree, and there are two partially scored characters that indicate a difference between A and C and A and D, respectively. The first three characters (taken together) do not support any tree, but the divergent character states from A to C and to D are easier to

explain on the AB tree. Note that if we ignore higher order terms (when p close to 0) the likelihood for the last two characters are:

$$\begin{aligned}
\Pr(0?1?|T_{AB}) = \Pr(0??1|T_{AB}) &\approx 3p && (1)\\
\Pr(0?1?|T_{AC}) &\approx 2p && (2)\\
\Pr(0??1|T_{AC}) &\approx 3p && (3)\\
\Pr(0?1?|T_{AD}) &\approx 3p && (4)\\
\Pr(0??1|T_{AD}) &\approx 2p && (5)
\end{aligned}$$

This reflects that fact that there are three branches that could display a change on the AB tree on characters that resemble a character shown in figure 1, but only two branches that provide an opportunity for a change for characters 2

Table 2: A data set for which ML under the equal-branch model prefers tree A+B, but MP does not prefer a tree. In the table, $p_{syn} = (1-p)^4p + 2(1-p)^3p^2 + p^5$, this is the probability of a synapomorphy that is compatible with the tree. The probability of a incompatible character pattern is $p_{inc} = 2(1-p)^3p^2 + 2(1-p)^2p^3$

		Characters				
Taxon	A	0	0	0	0	0
	B	0	1	1	?	?
	C	1	0	1	1	?
	D	1	1	0	?	1
	Tree					
Prob.	T_{AB}	p_{syn}	p_{inc}	p_{inc}	$3p(1-p)^2 + p^3$	$3p(1-p)^2 + p^3$
	T_{AC}	p_{inc}	p_{syn}	p_{inc}	$2p(1-p)$	$3p(1-p)^2 + p^3$
	T_{AD}	p_{inc}	p_{inc}	p_{syn}	$3p(1-p)^2 + p^3$	$2p(1-p)$

The example in table 2 is simple but artificial. It is tempting to think that, perhaps if we are given enough data then parsimony and ML under the equal branch length model will always converge to the same tree. In fact this is not the case (for general values of p). Kim (1996) shows an example of an (admittedly unrealistic) tree shape which has equal branch lengths. Given an unlimited amount of data parsimony recovers one tree (not the correct tree), but ML methods would recover the true tree.

Figure 1: A character that could be explained by one change on one of three branches.

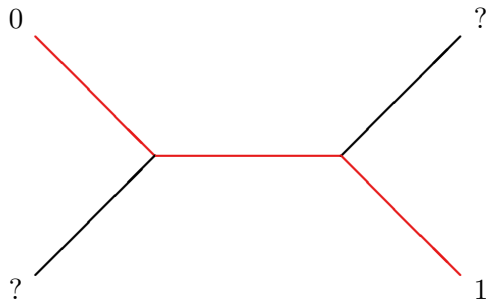
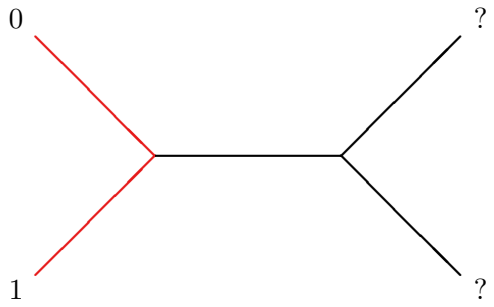


Figure 2: A character that could be explained by one change on one of two branches.



Goldman (1990) pointed out that if you use ML to infer not just the tree, but also the set of ancestral character states, then you will always prefer the same tree as parsimony. This amounts to using just one possible set of internal nodes assignments for each character.

If the parsimony length of character i is $s_i(T)$, then the reconstruction with the highest likelihood will be one of the reconstructions with the probability of $p^{s_i(T)}(1-p)^{2N-3-s_i(T)}$. The overall likelihood will be:

$$S(T) = \sum_{i=1}^{M} s_i(T) \tag{6}$$

$$\Pr(X|T) = p^{S(T)}(1-p)^{(2NM-3M-S(T))} \tag{7}$$

Because $0 < p < (1-p) < 1$ and N and M are constant across all trees, minimizing $S(T)$ will maximize the likelihood.

Table 3 shows the parsimony score for each character in Data matrix. Note that only the last three characters have scores that vary between the trees. These characters have scores shown in red. They are the ones that display potential synapomorphies within the ingroup, so they are the potentially informative characters in a Henngian character analysis. They are referred to as *parsimony-informative* because when we are using parsimony to evaluate trees, it is only characters like these that carry information for inferring the relationships between taxa.

Note that parsimony would prefer the tree that groups B+C together, because it has the smallest score (13) – the minimum number of character state changes required to explain the data. This is not too surprising: the B+C character had only one character (#12) for which we need to invoke homoplasy to explain the characters. In contrast, the B+D tree only requires one change for character #12, but requires two changes for characters #10 and #11. The C+D tree is the worst explanation, according to parsimony – no character shows a synapomorhy uniting C and D, and three characters (#10,#11, and #12) require multiple events to explain the character state distributions.

Table 3: Data

	Character #												
Taxon	1	2	3	4	5	6	7	8	9	10	11	12	
A	0	0	0	0	0	0	0	0	0	0	0	0	
B	1	0	0	0	0	1	1	1	1	1	1	1	
C	0	1	1	1	0	1	1	1	1	1	1	0	
D	0	0	0	0	1	1	1	1	1	0	0	1	
Tree	Character Score												Total Score
B+C	1	1	1	1	1	1	1	1	1	1	1	2	13
B+D	1	1	1	1	1	1	1	1	1	2	2	1	14
C+D	1	1	1	1	1	1	1	1	1	2	2	2	15

Fitch optimization

In the last section, I told you the minimal # of changes required for each character – but how did I know this number?

A brute force method would be to assign character states to all of the internal nodes of the tree. If we have a tree with internal states assignments, then the number of changes required is easy to calculate

A bit about sets

Walter Fitch's (1970 and 1971) algorithms for calculating the parsimony score of a tree use a bit of set notation. A **set** is a collection of objects. The objects are usually referred to as **elements** of the set. The common notation for a set is a pair of curly braces - {}.

The operations on sets used in Fitch's algorithm are the intersection and the union.

The **union** of two sets is a set that contains all of the elements that are in either sets. The union is denote $\cup$. For example:

$$\{2, 5, 6, 9\} \cup \{1, 5, 7, 9, 10\} = \{1, 2, 5, 6, 7, 9\}$$

The **intersection** of two sets is a set that contains only elements that are in *both* sets. The $\cap$ is used to represent an intersection. For example

$$\{2, 5, 6, 9\} \cap \{1, 5, 7, 9, 10\} = \{5, 9\}$$

An empty set, denote with a $\emptyset$, is a set that contains no elements. For example:

$$\{2, 6, 9\} \cap \{1, 5, 7, 10\} = \{\} = \emptyset$$

Fitch down-pass

Walter Fitch showed that you can calculate the best possible parsimony score for a character on a tree (the fewest number of changes-of-state that explain the data), using a single pass down the tree from the leaves to the root.

Algorithm 1 Fitch Down Pass

score = 0
while There are still unvisited nodes **do**
  choose a node, a, that has not been visited, but all of the node's children must be leaves or nodes that have already been visited.
  Let d_a denote the down-pass state set of the ancestral node. Since we did not observe this species, d_a is unknown.
  Let d_L denote the state set of the left descendant (this will be known or inferred in a previous round).
  Let d_R denote the state set of the right descendant (this will be known or inferred in a previous round).
  $d_i = d_L \cap d_R$
  if $d_i = \emptyset$ **then**
    $d_a = d_L \cup d_R$
    add 1 to the score
  else
    $d_a = d_L \cap d_R$
  end if
end while

Fitch's down pass algorithm is shown in a box. You can think of the logic of the algorithm as:

1. "if the two descendants show the same states, then assume that the ancestor had the states that are in common" – this is the part in which you assign $d_A = d_L \cap d_R$ if $d_L \cap d_R \neq \emptyset$.
2. "if the two descendants have different states, then assume that the ancestor could have had *any* state that is in at least one of the children. If this is the case, then we know that we will need to have one change-of-state in this part of the tree, so we add one to the score" – this is the part in which we augment the score, and assign $d_A = d_L \cup d_R$.

Important note: this down-pass of the Fitch algorithm only gives us the parsimony score of the character on the tree. It does not (necessarily) give us the most-parsimonious ancestral character state reconstructions for each node. Technically the ancestral state sets are referred to as the *preliminary* state sets for the nodes.

To evaluate trees using parsimony we do not need to know all of the ancestral character state assignments, we just need the score. So the "down-pass" of Fitch's algorithm gives us what we need to score a tree using parsimony.

Figures 3-11 give an example of how the down-pass of the Fitch algorithm works.

Fitch up-pass

If we do want to know the possible most parsimonious character state reconstructions, then we have to complete an up-pass. The up-pass algorithm is shown in a box, and figures 12-19 give an example of how the up-pass of the Fitch algorithm works.

Fitch's up pass gives the final state set at each node – the list of possible states that exist in any most parsimonious reconstruction. But, if we want to know what changes exist in a most parsimonious reconstruction, we can **not** simply consider all possible connections between states that are in the final (up-pass) sets. Once again, Fitch (1971) provides an algorithm

Algorithm 2 Fitch Up Pass

start at the root
make the root's up-pass set identical to the down-pass set: $u_r = d_r$
while There are still unvisited nodes **do**
    choose the node n that is closest to the root, but has not been visited yet.
    Let u_n and denote the up-pass state set of a node n.
    Let d_n denote the down-pass state set of node n.
    Let u_p denote the up-pass state set of the node n's parent node.
    Let d_L and d_R denote the down-pass state sets of node n's children
    if $d_n \cap u_p = u_p$ **then**
        $u_n = d_n \cap u_p$
    else if $d_L \cap d_R = \emptyset$ **then**
        $u_n = d_n \cup u_p$
    else
        $u_n = d_n \cup (u_p \cap (d_L \cup d_R))$
    end if
end while

Algorithm 3 Most parsimonious changes across branches

Require: Perform the Fitch Down Pass and then the Fitch Up Pass for the tree. Then for any edge e you can use this algorithm.

let c be the child node of the edge
let p be the parent node of the edge
for each state s in the up-pass set of p (for each $s \in u_p$) **do**
    if s is in the down-pass set of c (if $s \in d_c$) **then**
        $s \rightarrow s$ is a state-to-state transition across branch e in a MPR.
    else
        for each state t in the down-pass set of c (for each $t \in d_c$) **do**
            $s \rightarrow t$ is a state-to-state transition across branch e in a MPR.
        end for
        if s is in the up-pass set of c (if $s \in u_c$) **then**
            $s \rightarrow s$ is a state-to-state transition across branch e in a MPR.
        end if
    end if
end for

Examples of various steps of Fitch's algorithms

Table 4: A tree and Fitch's down-pass and up-pass algorithms. Step numbers refer to the number of the steps in the "Fitch Up Pass" box. This figure demonstrates the need for steps 9 and 11 of the algorithm (note that the down-pass set for the two internals is not complete and the up-pass is needed to reveal the full set of possible mpr).

G	Down-pass	Up-pass	Up-pass step #
C —	$\{C, G\}$	$\{A, C, G\}$	step # 11
A —	$\{A, C, G\}$	$\{A\}$	step # 9
A —	$\{A\}$	$\{A\}$	step # 2

Table 5: A tree and Fitch's down-pass and up-pass algorithms. Step numbers refer to the number of the steps in the "Fitch Up Pass" box. This figure demonstrates a case in which step 13 is used, but does not add any new states.

G	Down-pass	Up-pass	Up-pass step #
G —	$\{G\}$	$\{G\}$	step # 13, but $u_p \cap (d_L \cup d_R) = \emptyset$
A —	$\{A, G\}$	$\{A\}$	step # 9
A —	$\{A\}$	$\{A\}$	step # 2

Table 6: A tree and Fitch's down-pass and up-pass algorithms. Step numbers refer to the number of the steps in the "Fitch Up Pass" box. This figure demonstrates a case in which step 13 is used and the step adds a new state to a node's state set.

G	Down-pass	Up-pass	Up-pass step #
A —	$\{A, G\}$	$\{A, G\}$	step # 9
G —	$\{G\}$	$\{A, G\}$	step # 13, and $u_p \cap (d_L \cup d_R) = \{A\}$
A —	$\{A, G\}$	$\{A\}$	step # 9
A —	$\{A\}$	$\{A\}$	step # 2

Table 7: A tree and Fitch's down-pass, up-pass, and most-parsimonious changes algorithms. Step numbers refer to the number of the steps in the "Fitch Up Pass" box. Following Fitch, the states added to the state set during the up-pass are denoted in with (). The allowed changes on all terminal branches are omitted. These reconstructions are simply the up-pass set → the state show for the leaf).

G	Down-pass	Up-pass	Most parsimonious branch reconstructions.
A —	$\{A, G\}$	$\{A, G\}$	
			$A \rightarrow A$ and $G \rightarrow G$
G —	$\{G\}$	$\{(A), G\}$	
			$A \rightarrow \{A, G\}$ and $\{C, G\} \rightarrow G$
C —	$\{C, G\}$	$\{(A), C, G\}$	
			$\{A\} \rightarrow \{A, C, G\}$ and $C \rightarrow C$
A —	$\{A, C, G\}$	$\{A, C\}$	
			$A \rightarrow A$ and $C \rightarrow C$
C —	$\{C\}$	$\{(A), C\}$	
			$A \rightarrow \{A, C\}$ and $C \rightarrow C$
A —	$\{A, C\}$	$\{A, C\}$	

Detailed example of Fitch's algorithms

Table 8: A data matrix consisting of one character in a DNA sequence

Taxon	Character state
A	A
B	C
C	C
D	A
E	A
F	C
G	G
H	A
I	A

Figure 3: Fitch algorithm steps 1 and 2 for that DNA sequence character shown in Table 8

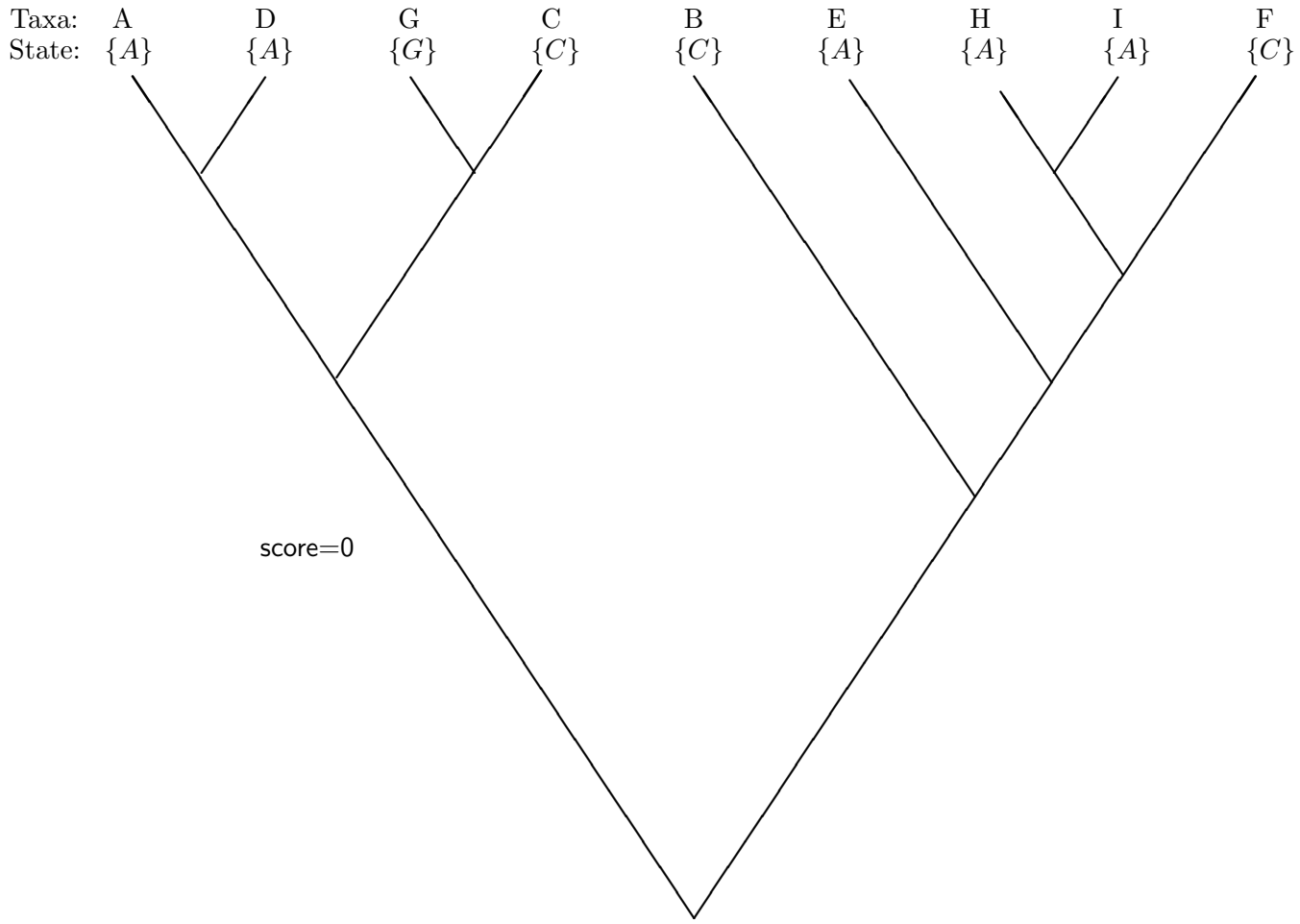


Figure 4: Fitch algorithm step 3 (first ancestral node) for that DNA sequence character shown in Table 8

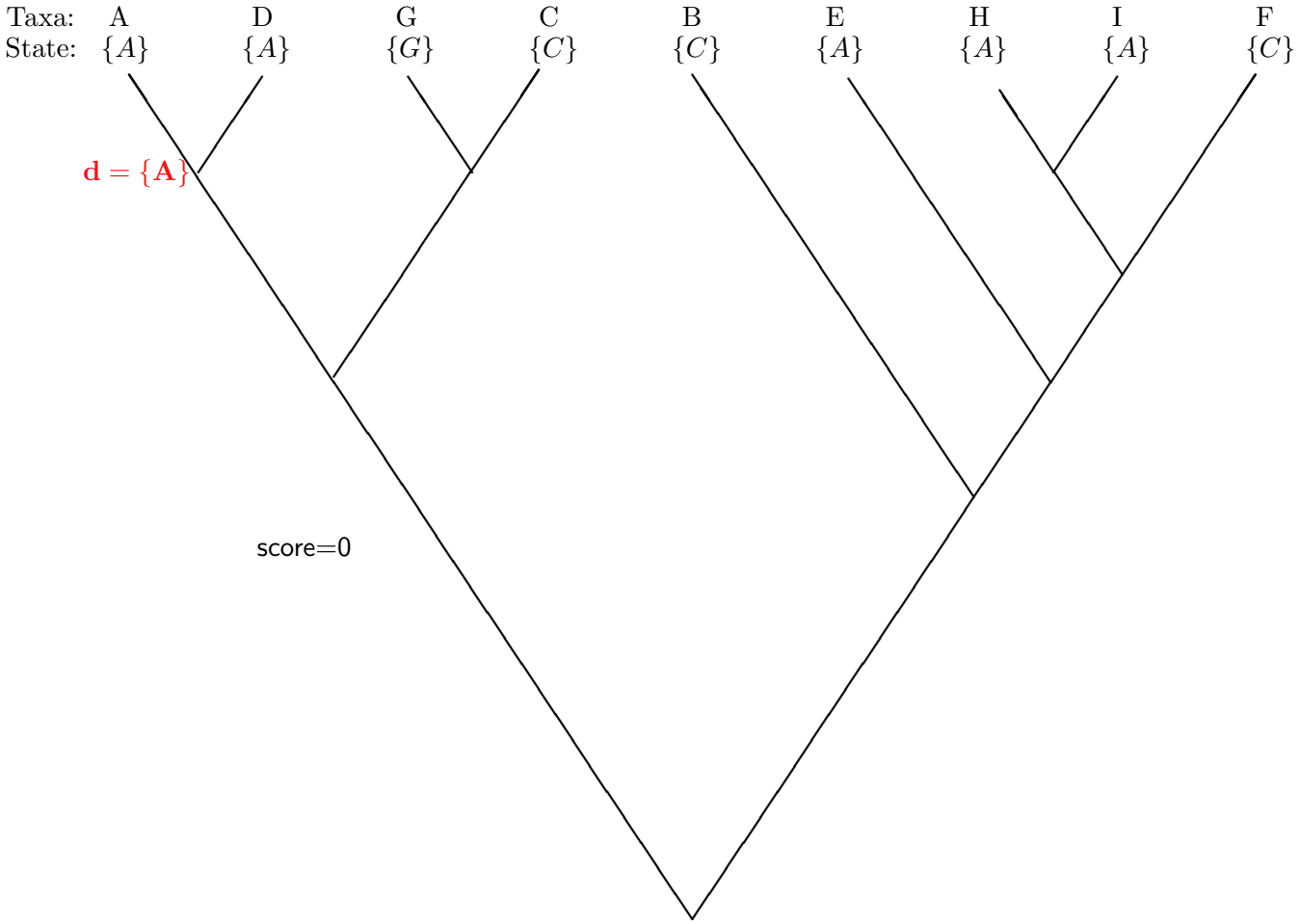


Figure 5: Fitch algorithm step 3 (second ancestral node) for that DNA sequence character shown in Table 8

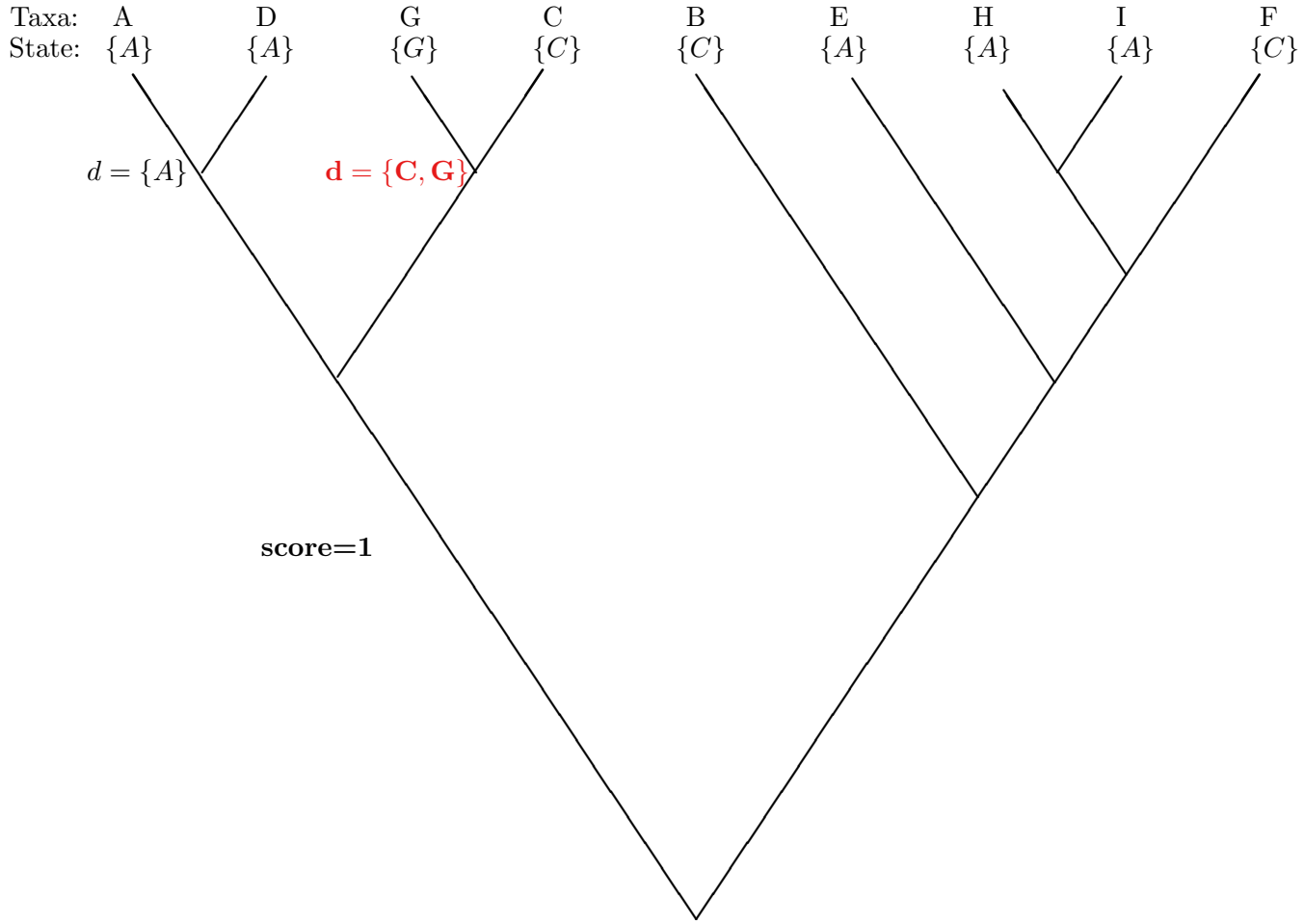


Figure 6: Fitch algorithm step 3 (third ancestral node) for that DNA sequence character shown in Table 8

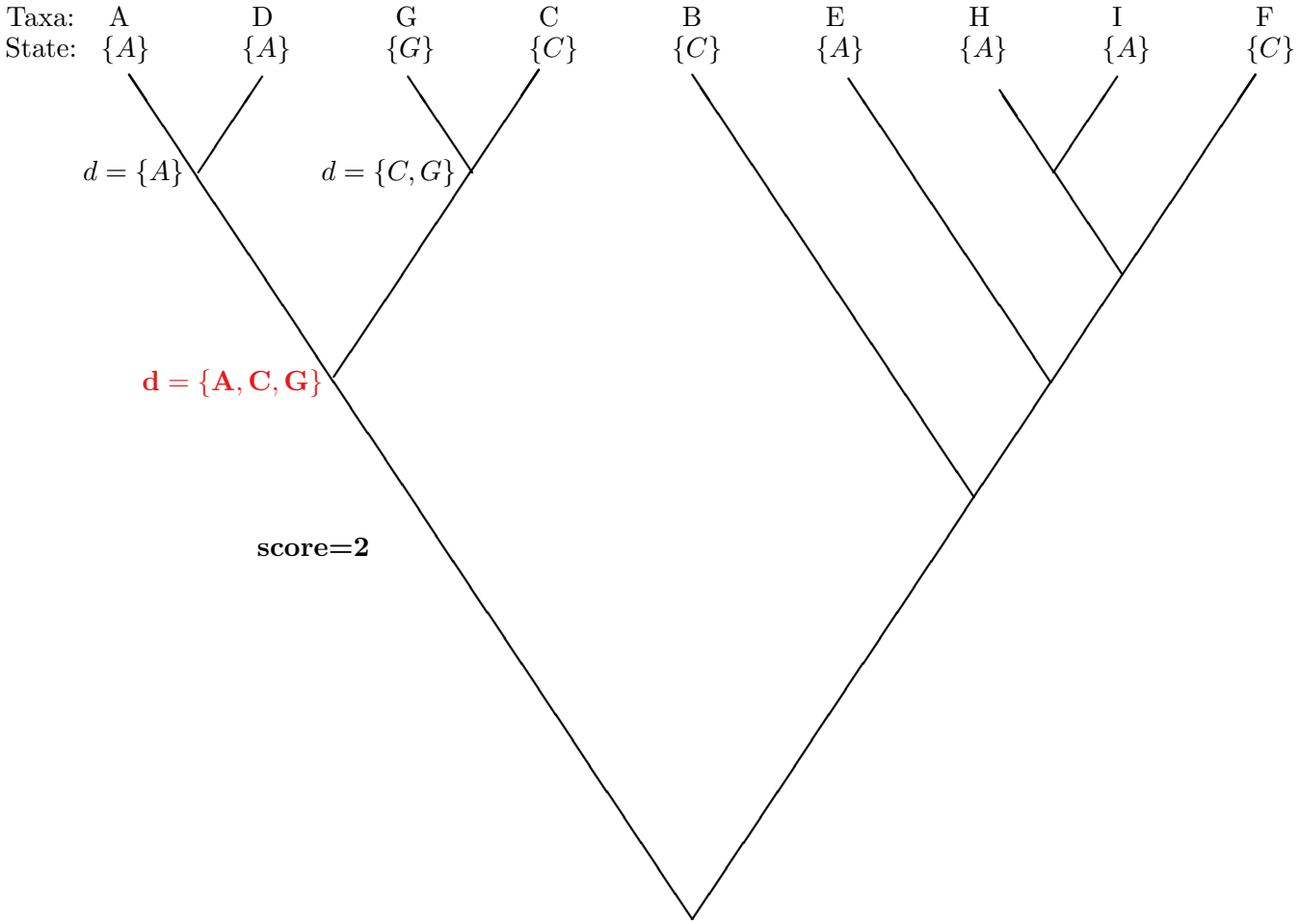


Figure 7: Fitch algorithm step 3 (fourth ancestral node) for that DNA sequence character shown in Table 8

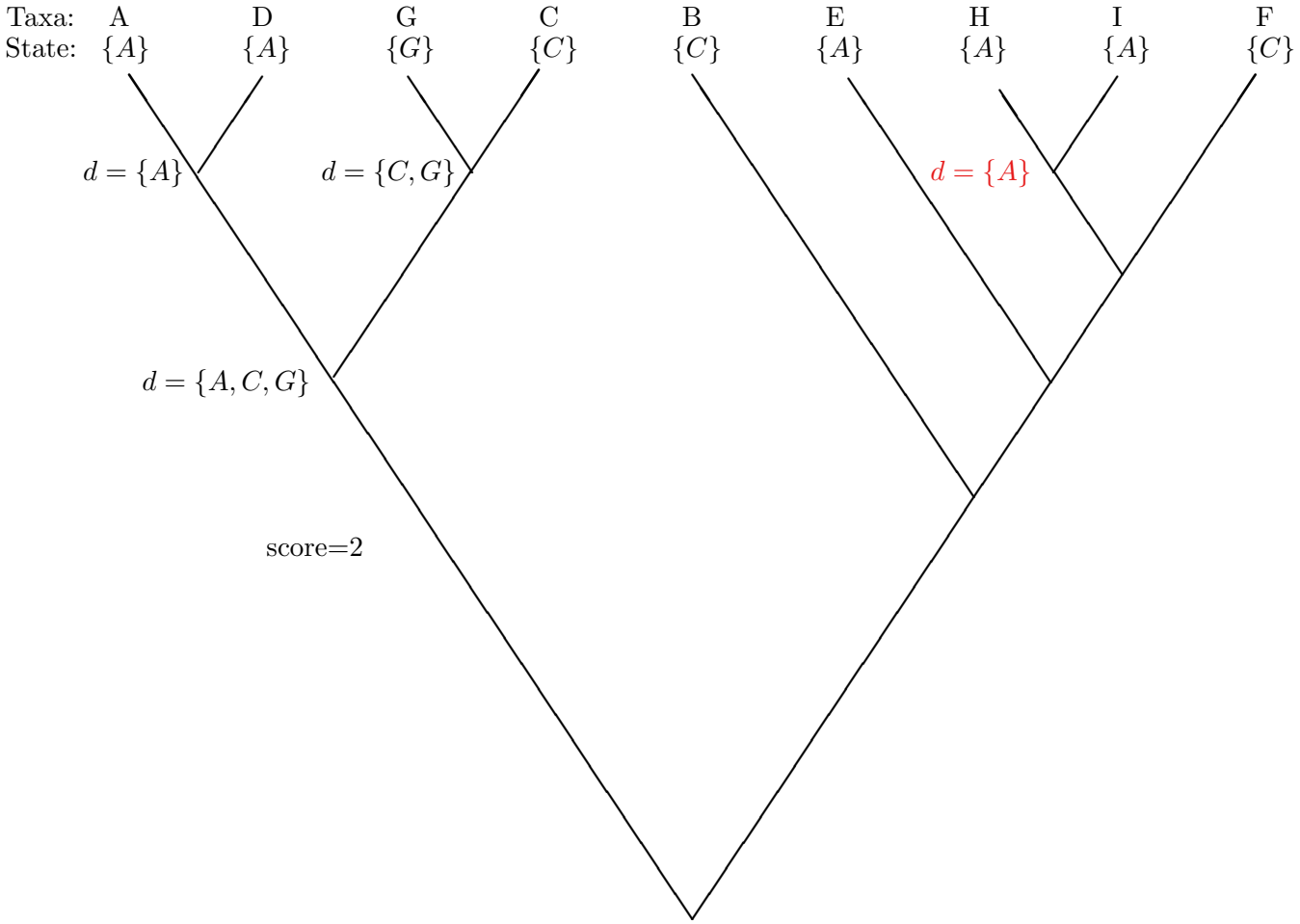


Figure 8: Fitch algorithm step 3 (fifth ancestral node) for that DNA sequence character shown in Table 8

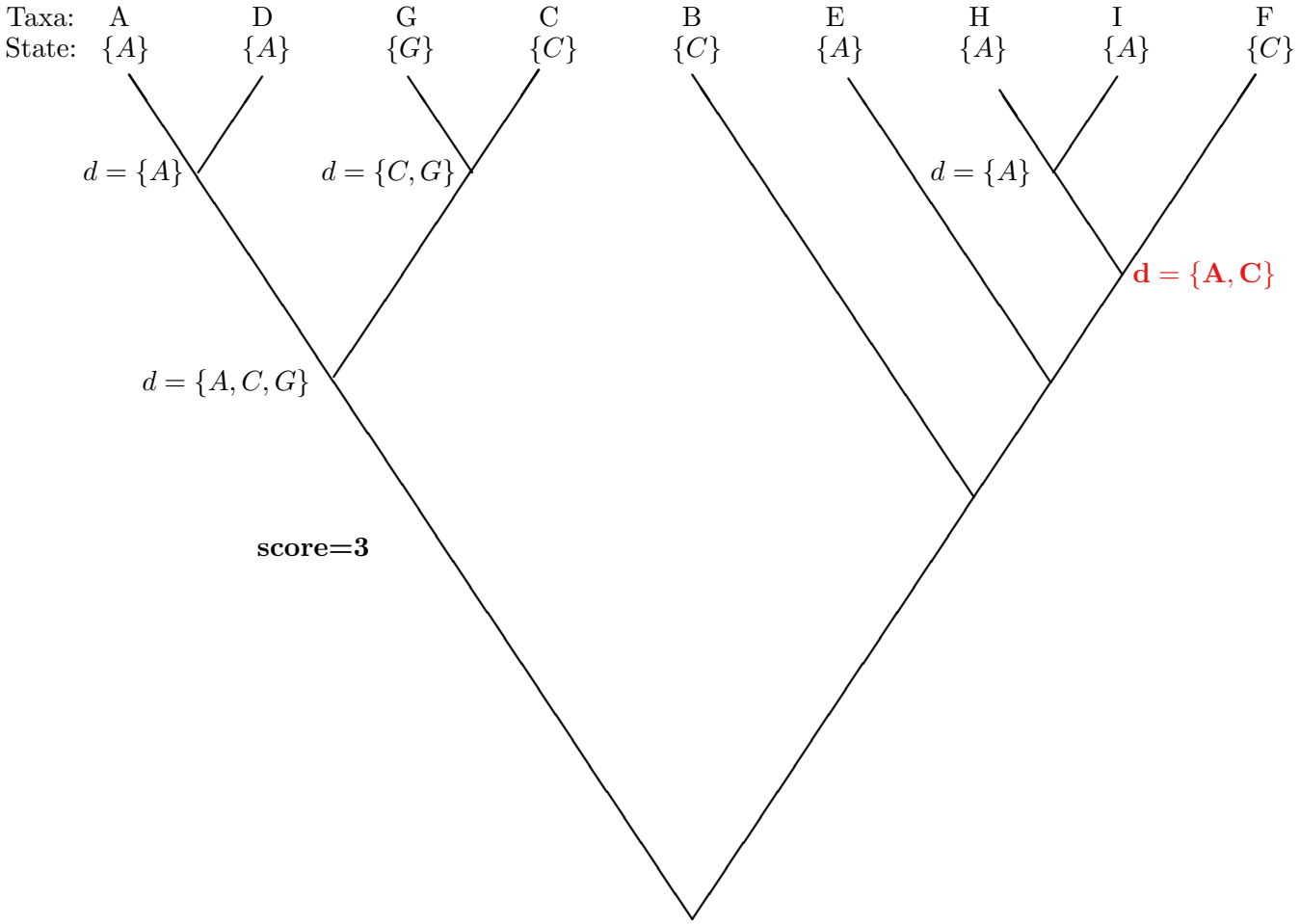


Figure 9: Fitch algorithm step 3 (fifth ancestral node) for that DNA sequence character shown in Table 8

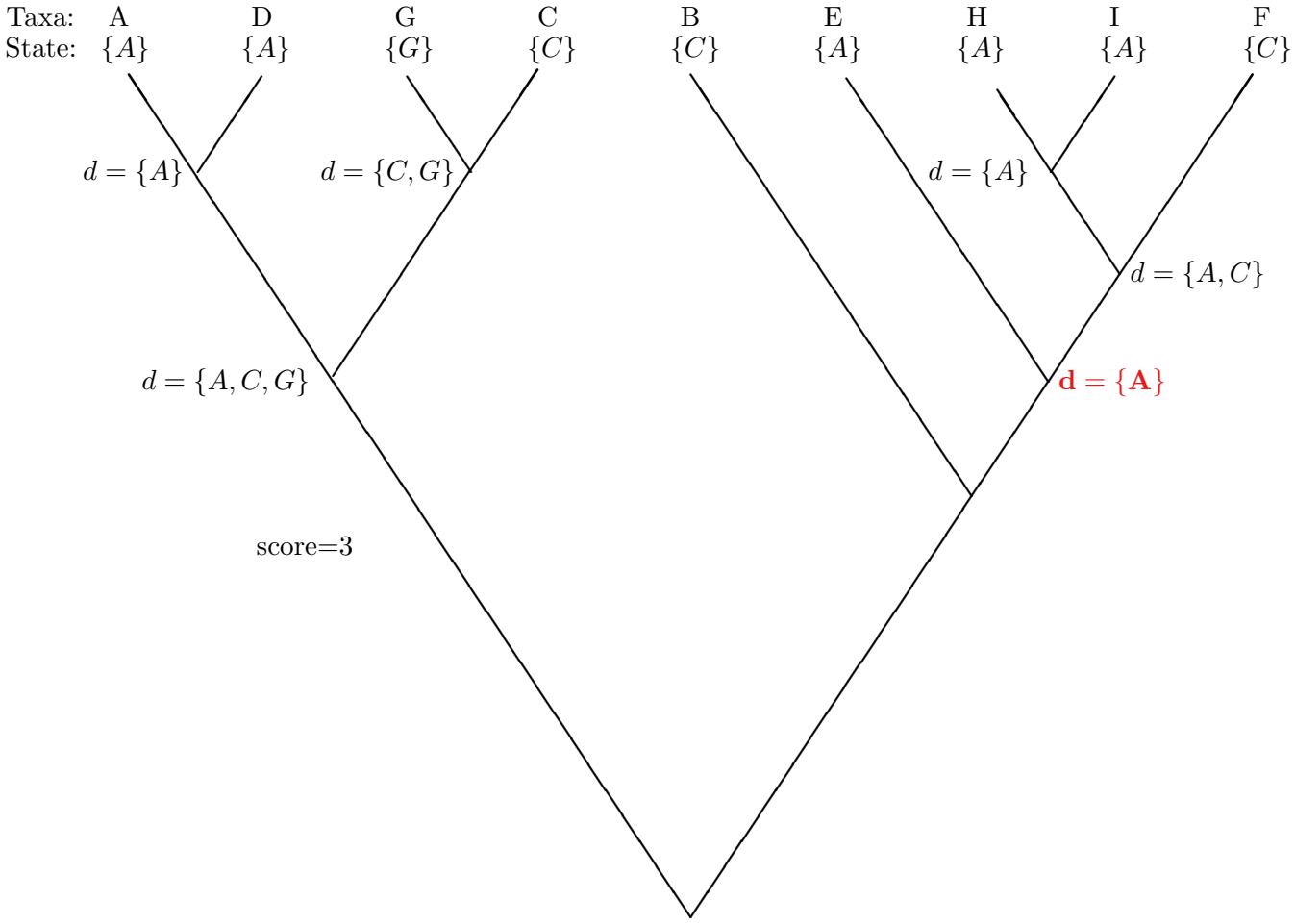


Figure 10: Fitch algorithm step 3 (sixth ancestral node) for that DNA sequence character shown in Table 8

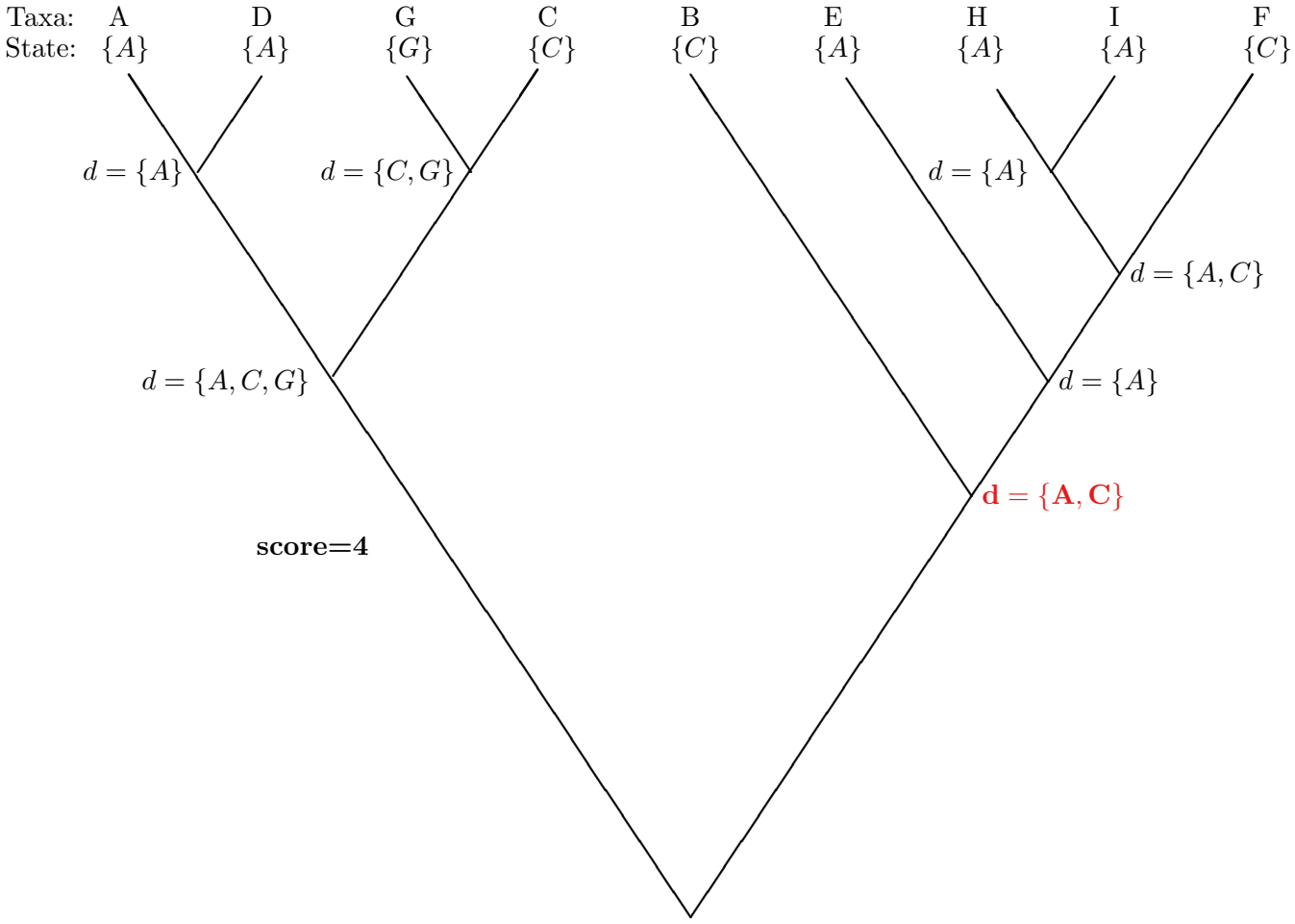


Figure 11: Fitch algorithm step 3 (seventh ancestral node – root) for that DNA sequence character shown in Table 8

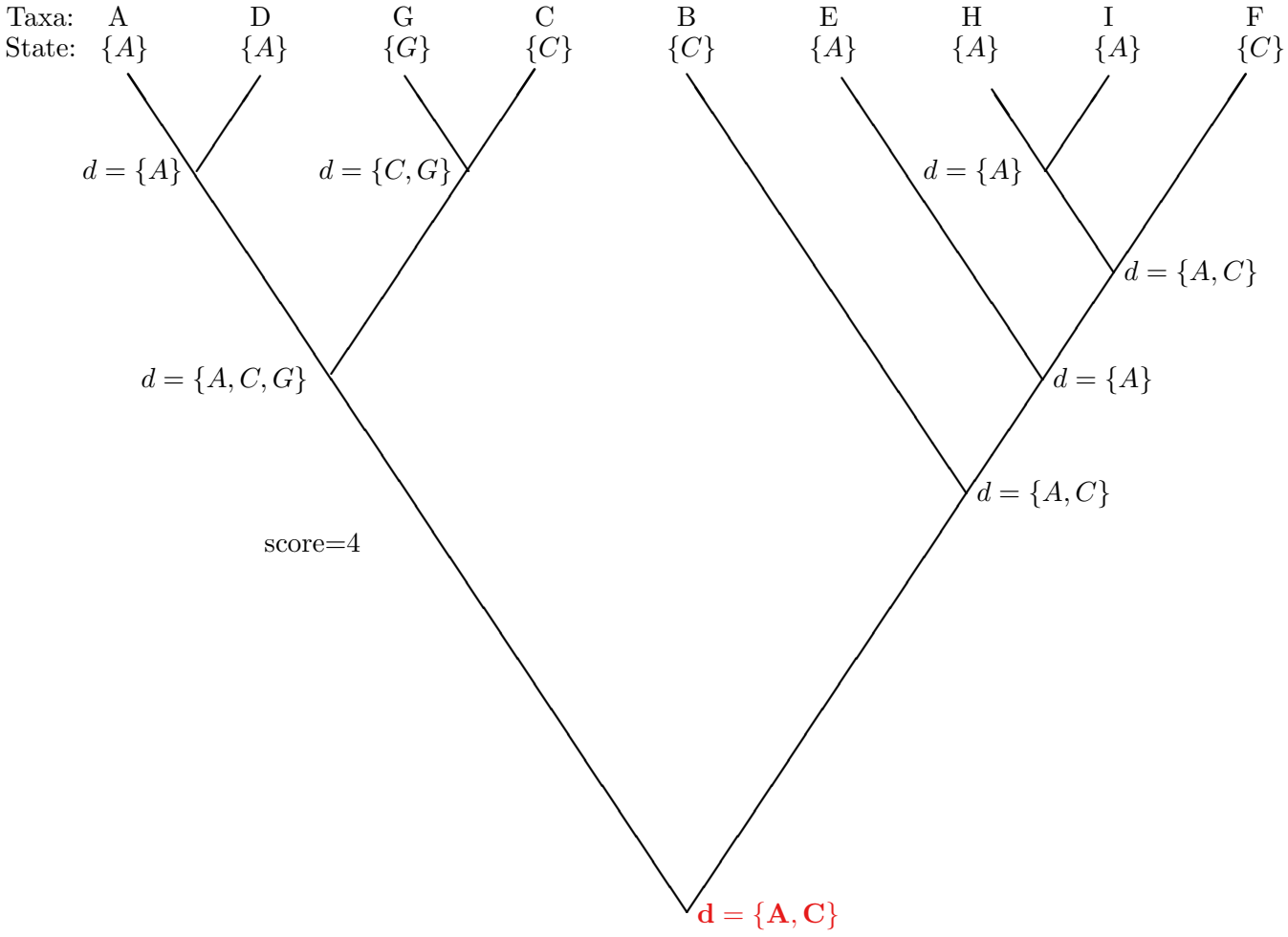


Figure 12: Fitch up-pass at root

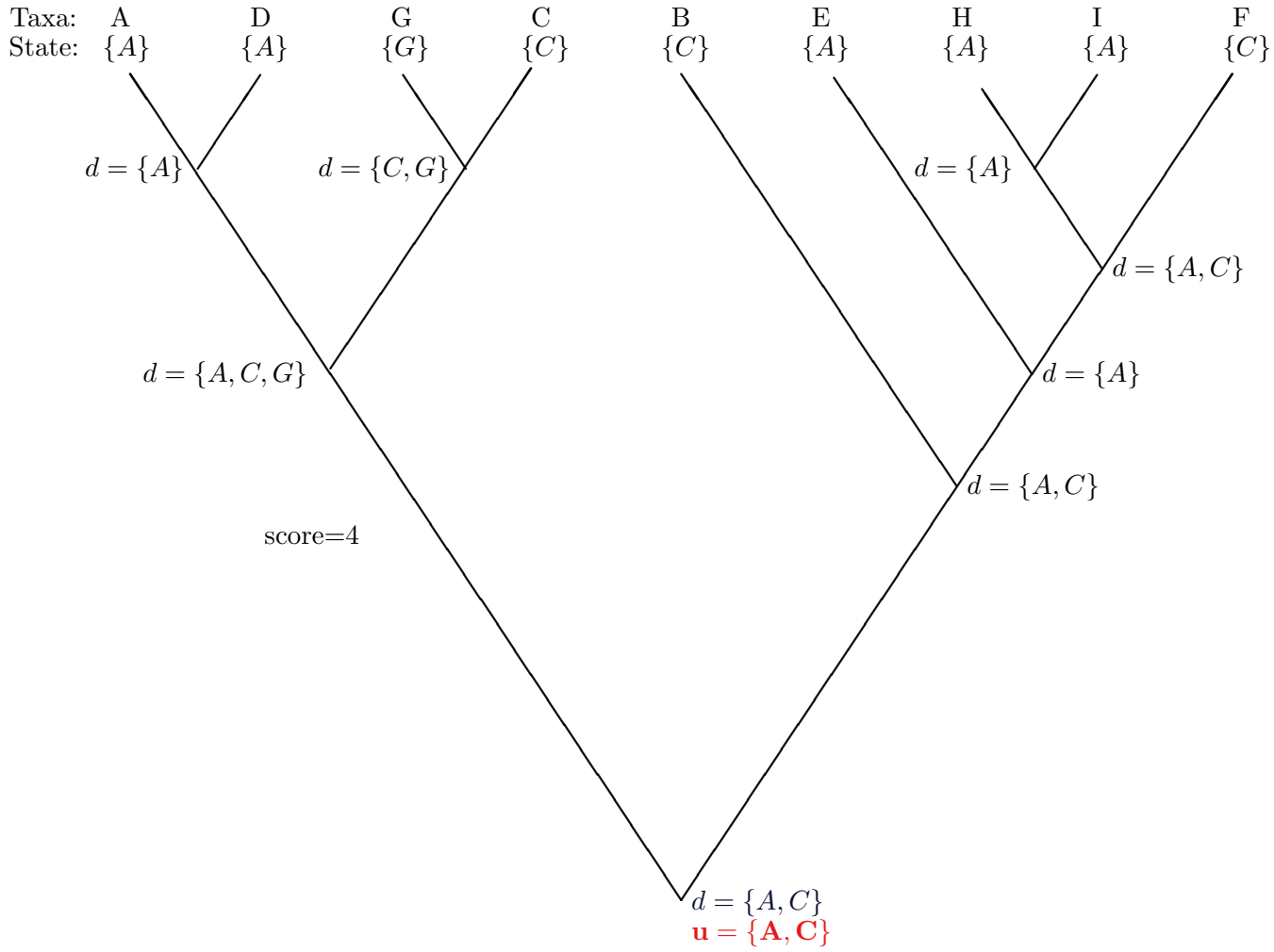


Figure 13: Fitch up-pass second level nodes

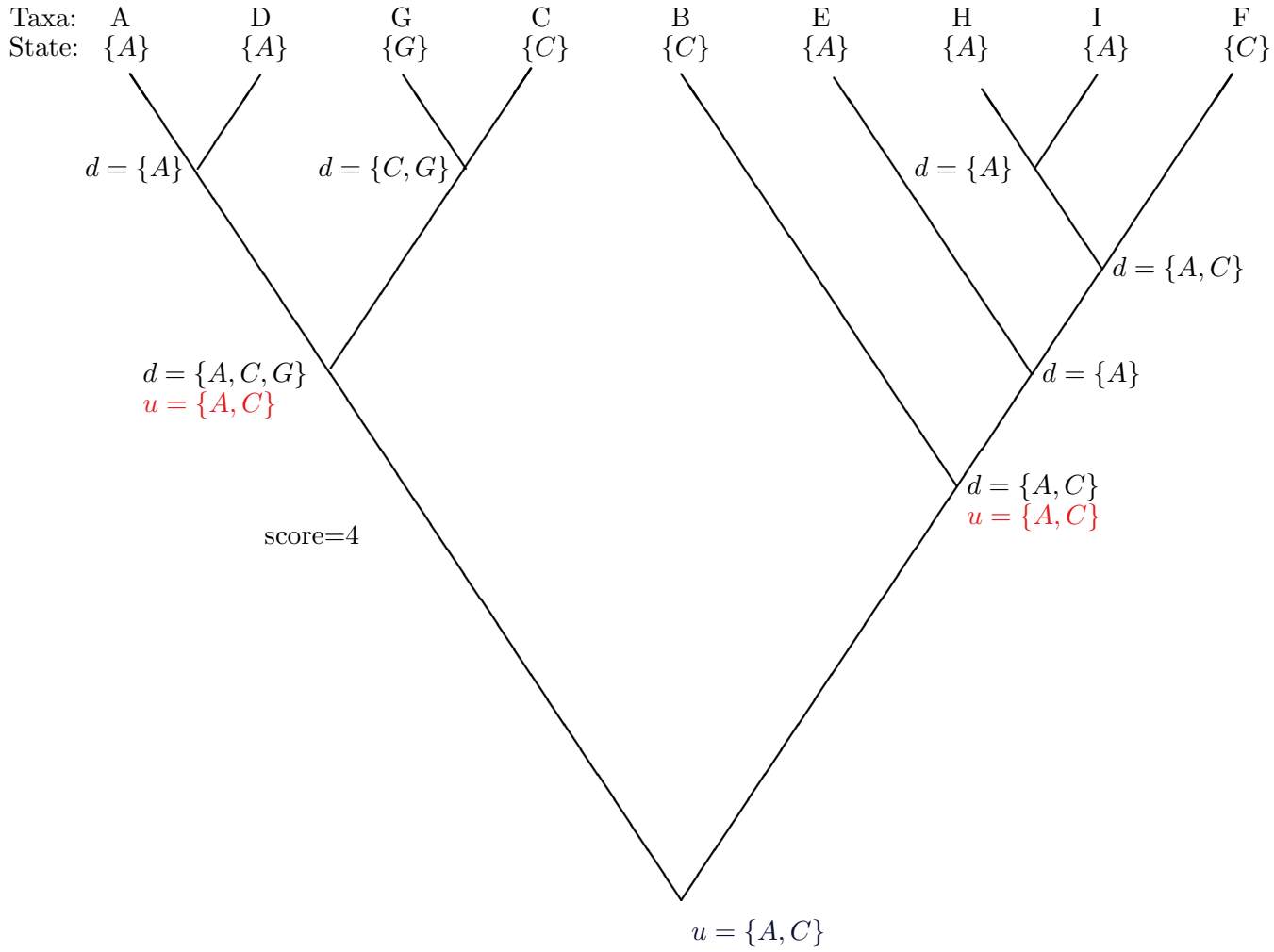


Figure 14: Fitch up-pass third level nodes

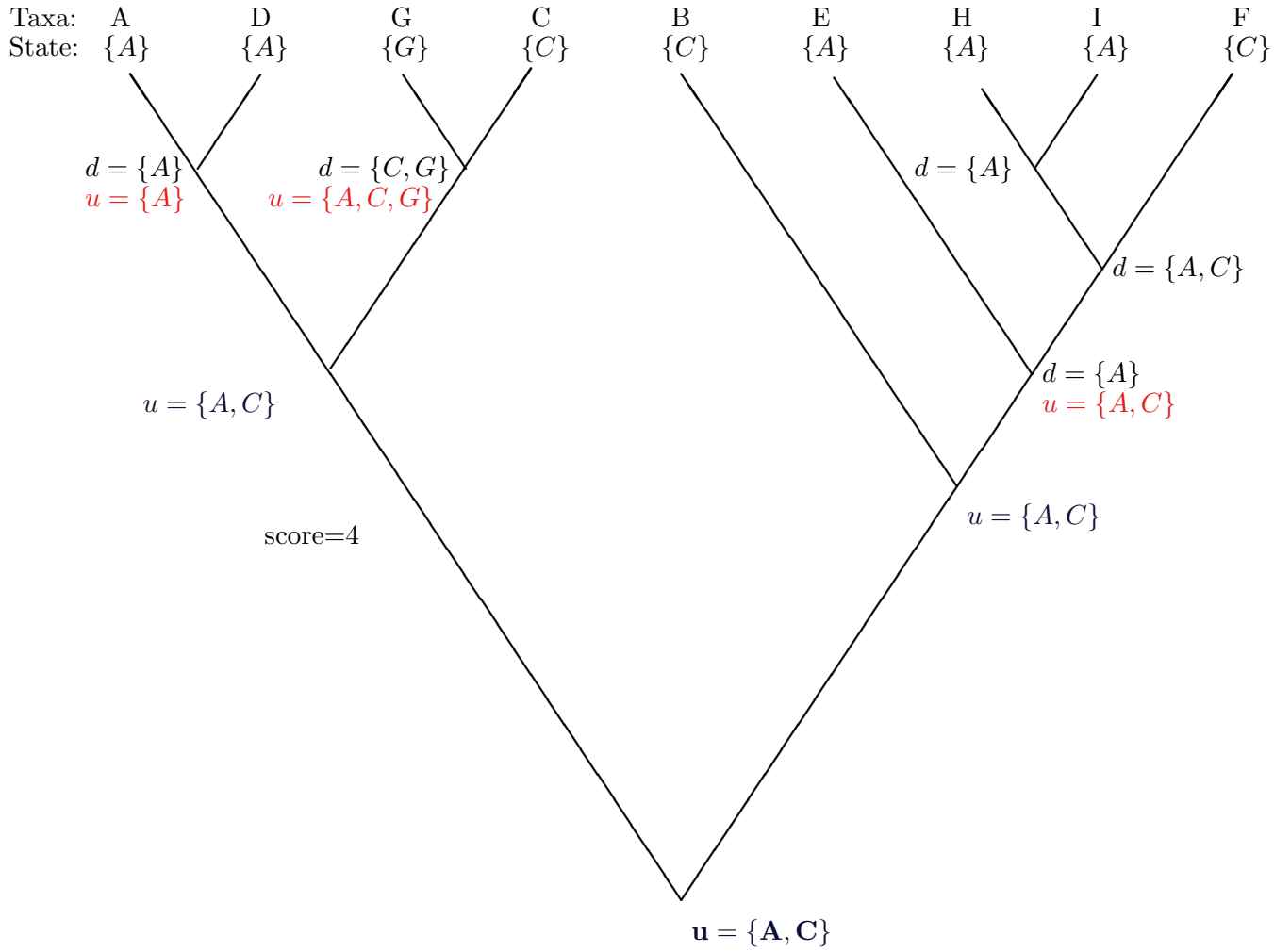


Figure 15: Fitch up-pass fourth level nodes

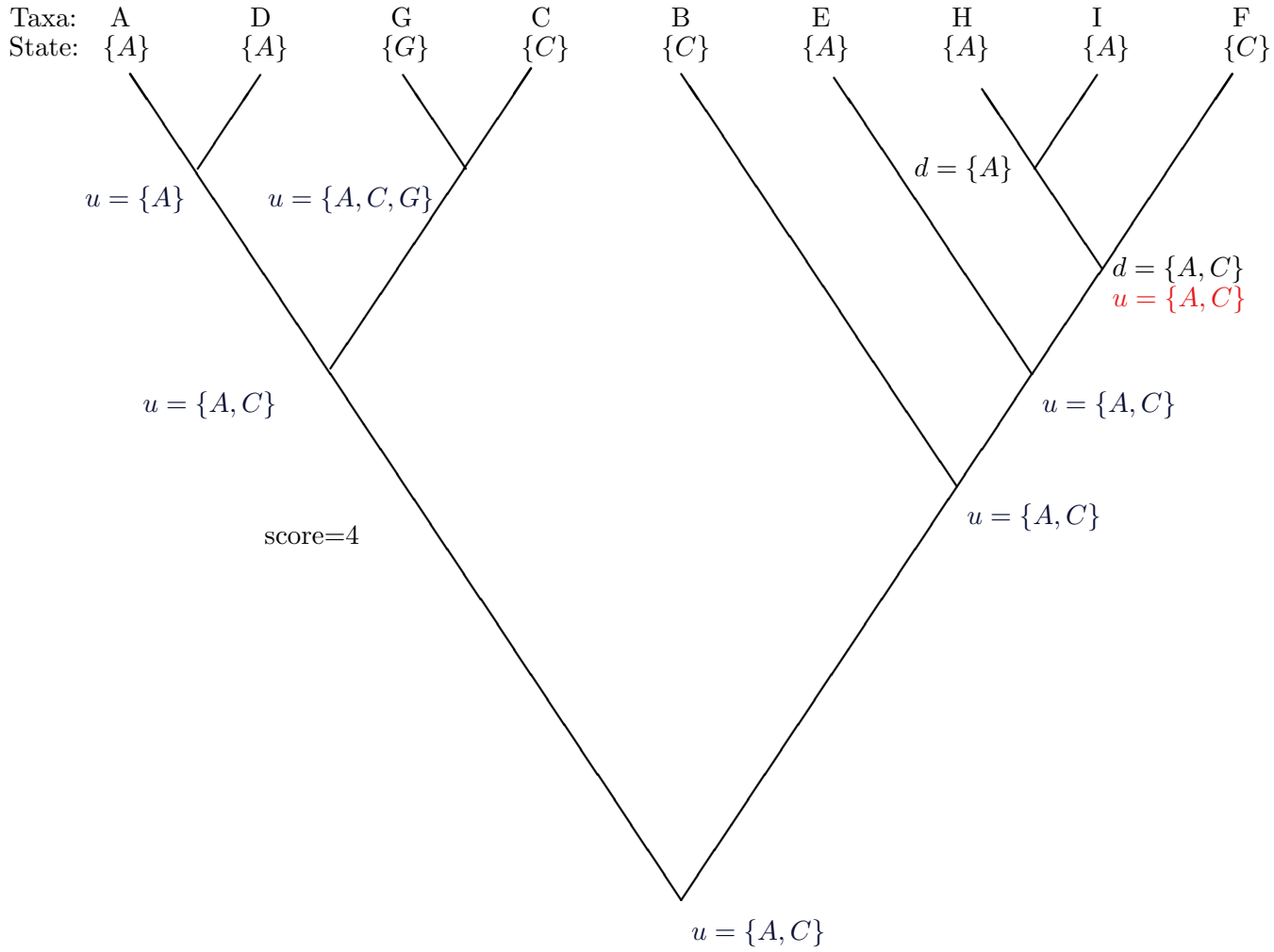


Figure 16: Fitch up-pass fifth level nodes

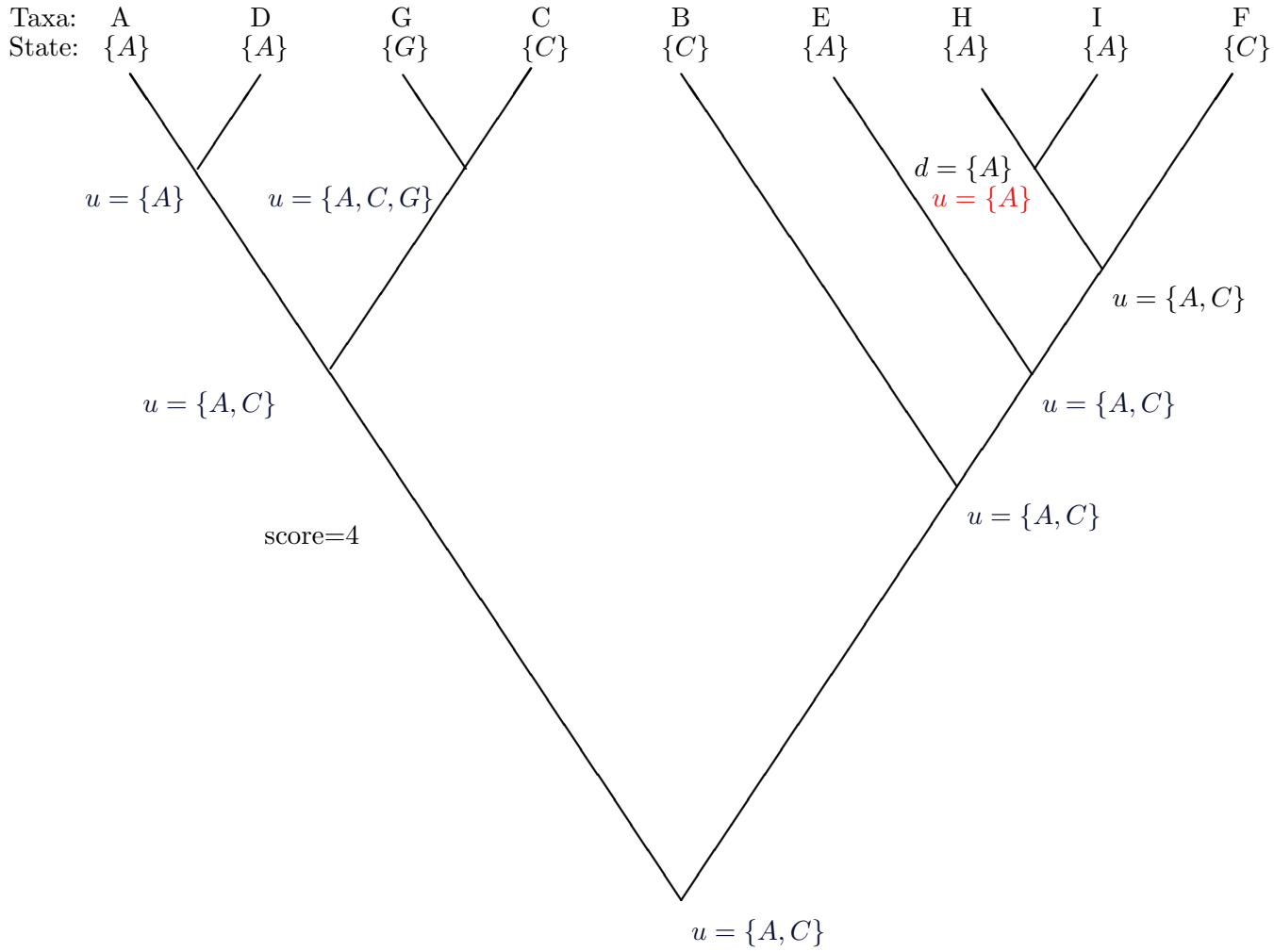


Figure 17: Fitch after up-pass

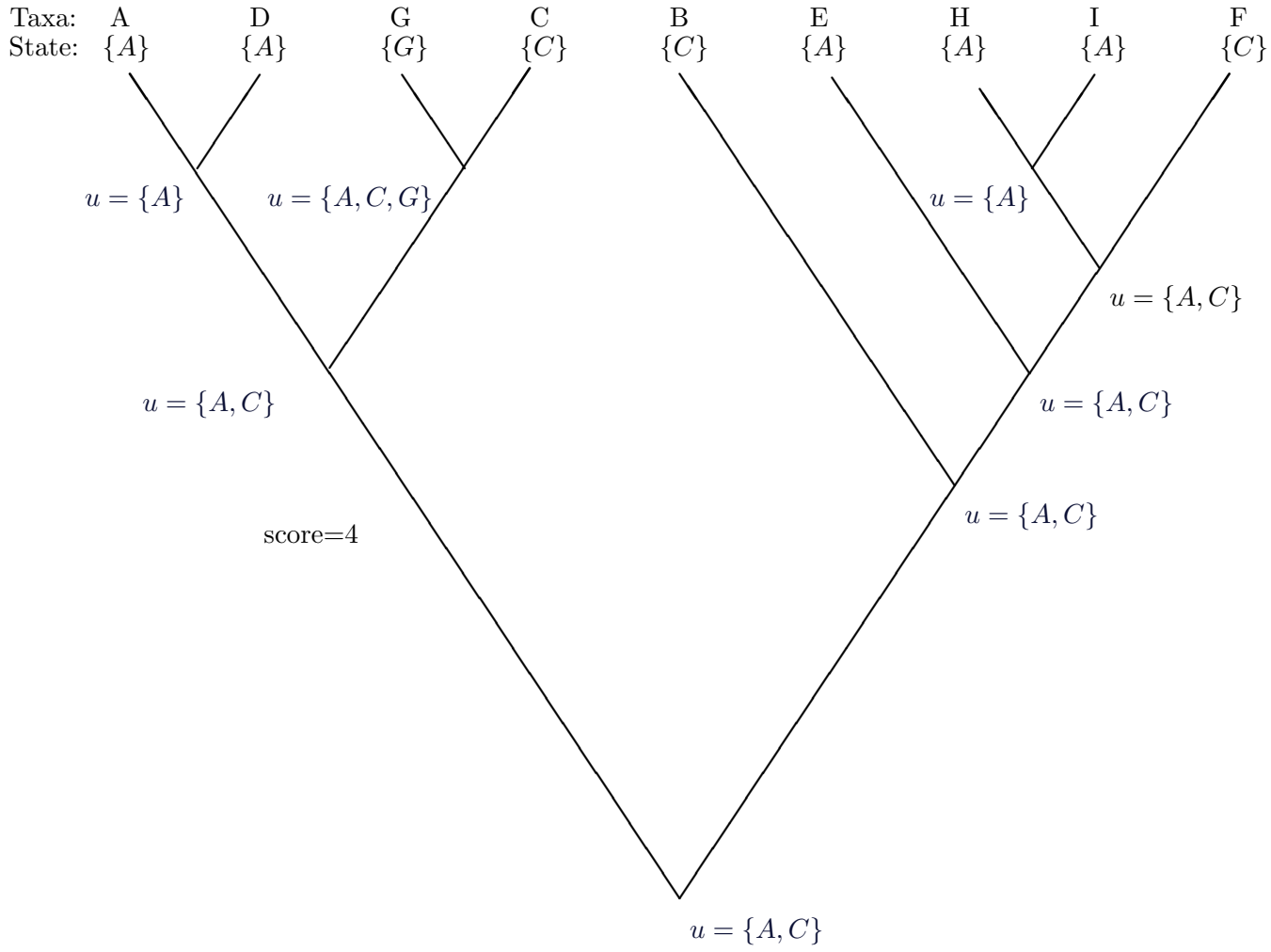


Figure 18: Fitch final state set with states added in up-pass shown in ()

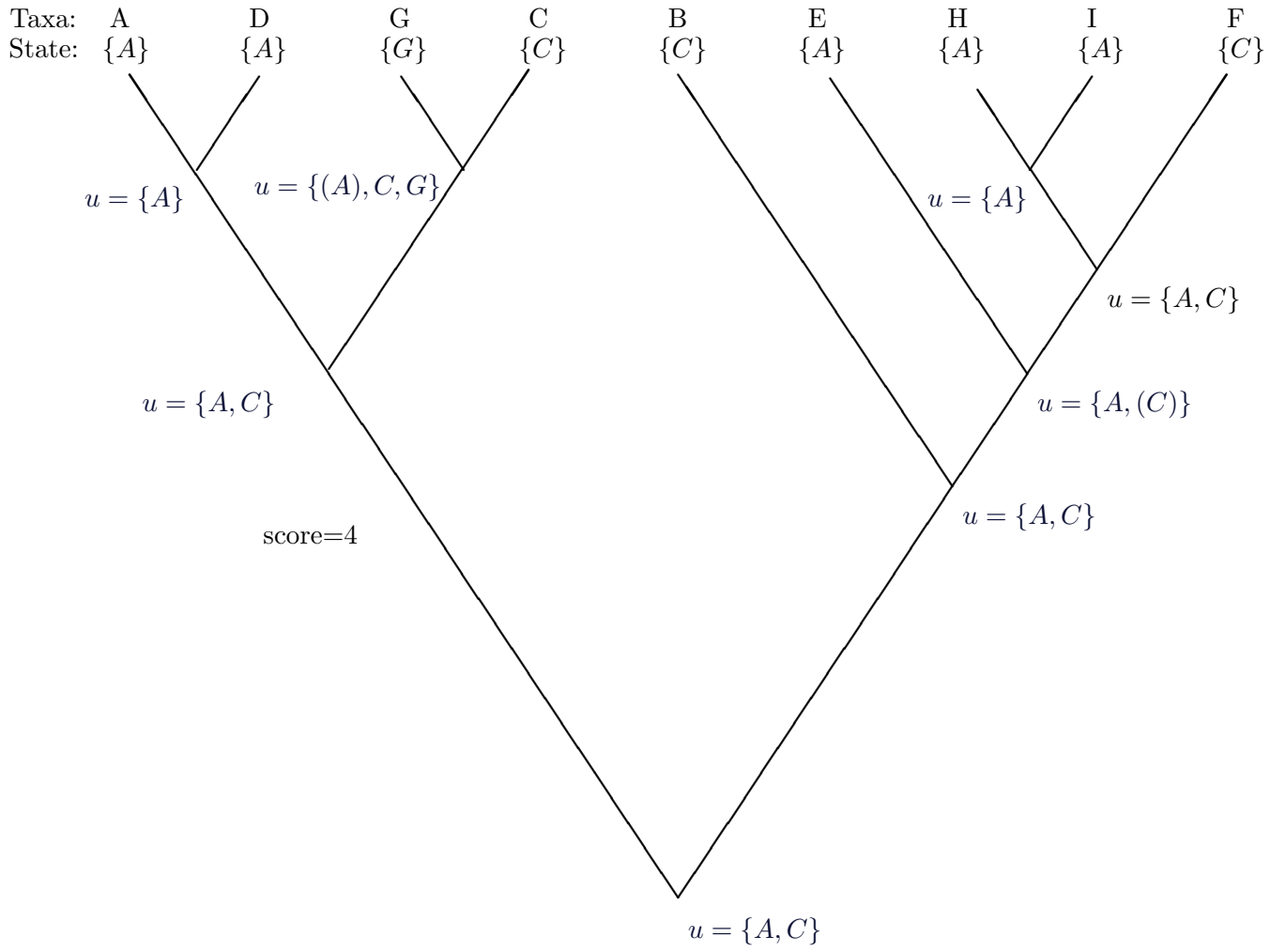
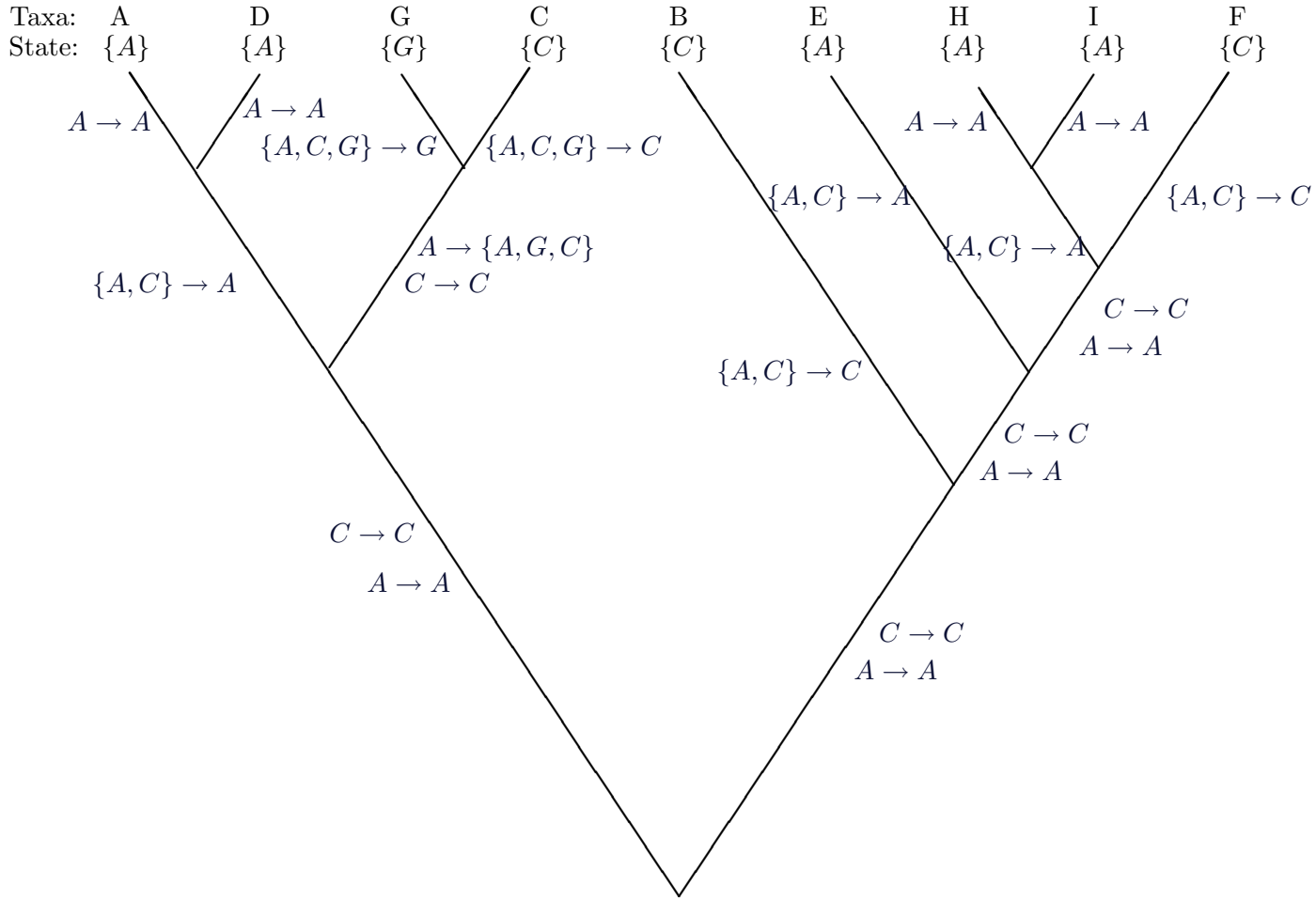


Figure 19: transitions in mpr



References

Fitch, W. M. (1971). Toward defining the course of evolution: Minimum change for a specified tree topology. *Systematic Zoology*, 20:406–416.

Goldman, N. (1990). Maximum likelihood inference of phylogenetic trees, with special reference to a poisson process model of DNA substitution and to parsimony analyses. *Systematic Zoology*, 39(4):345–361.

Kim, J. (1996). General inconsistency conditions for maximum parsimony: Effects of branch lengths and increasing number of taxa. *Systematic Biology*, 45:363–374.