RECONCILIATION BETWEEN GENE TREES AND SPECIES TREES IN THE PHYLOGENOMICS ERA

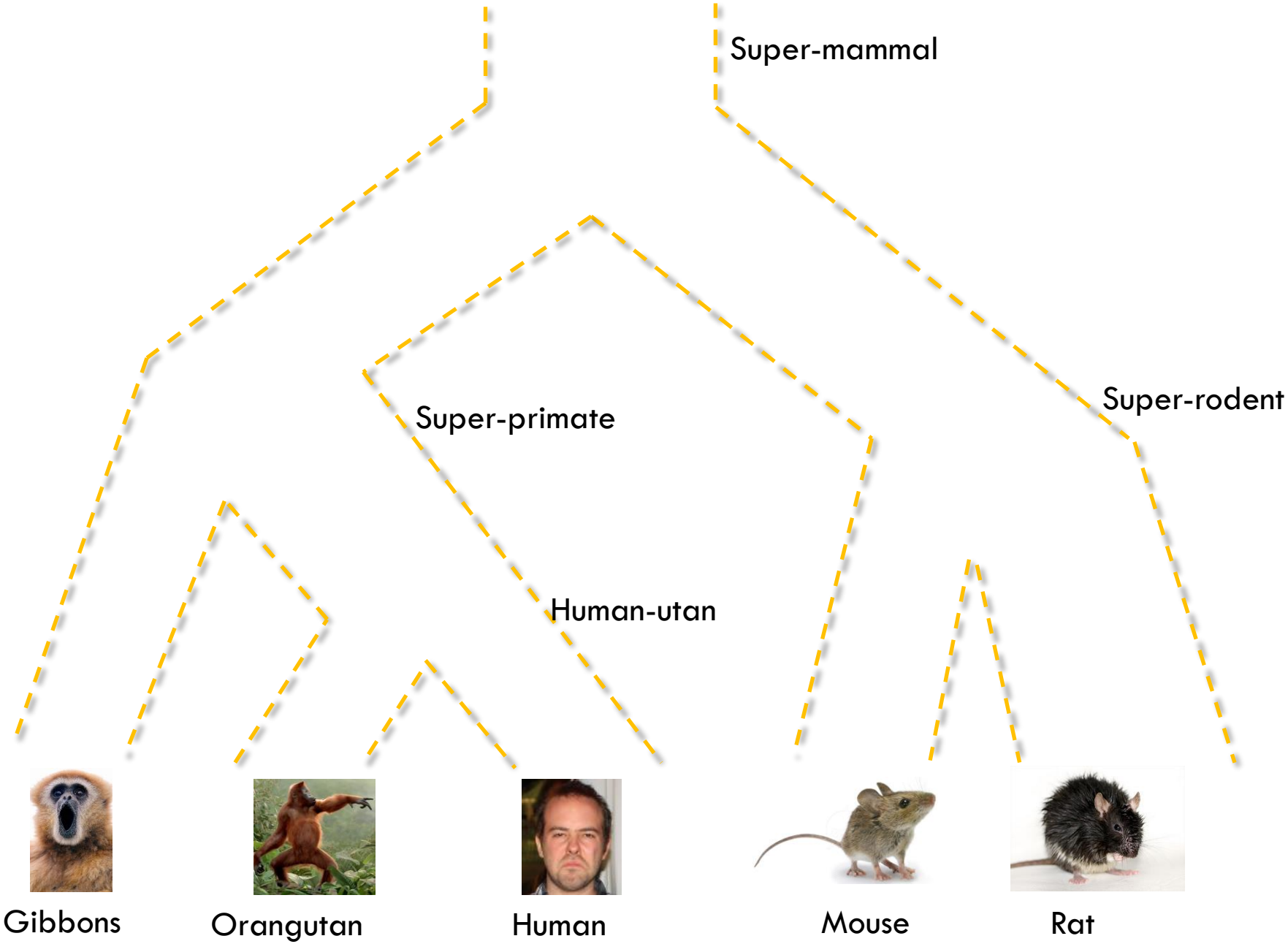
Manuel Lafond
Université de Sherbrooke, Canada

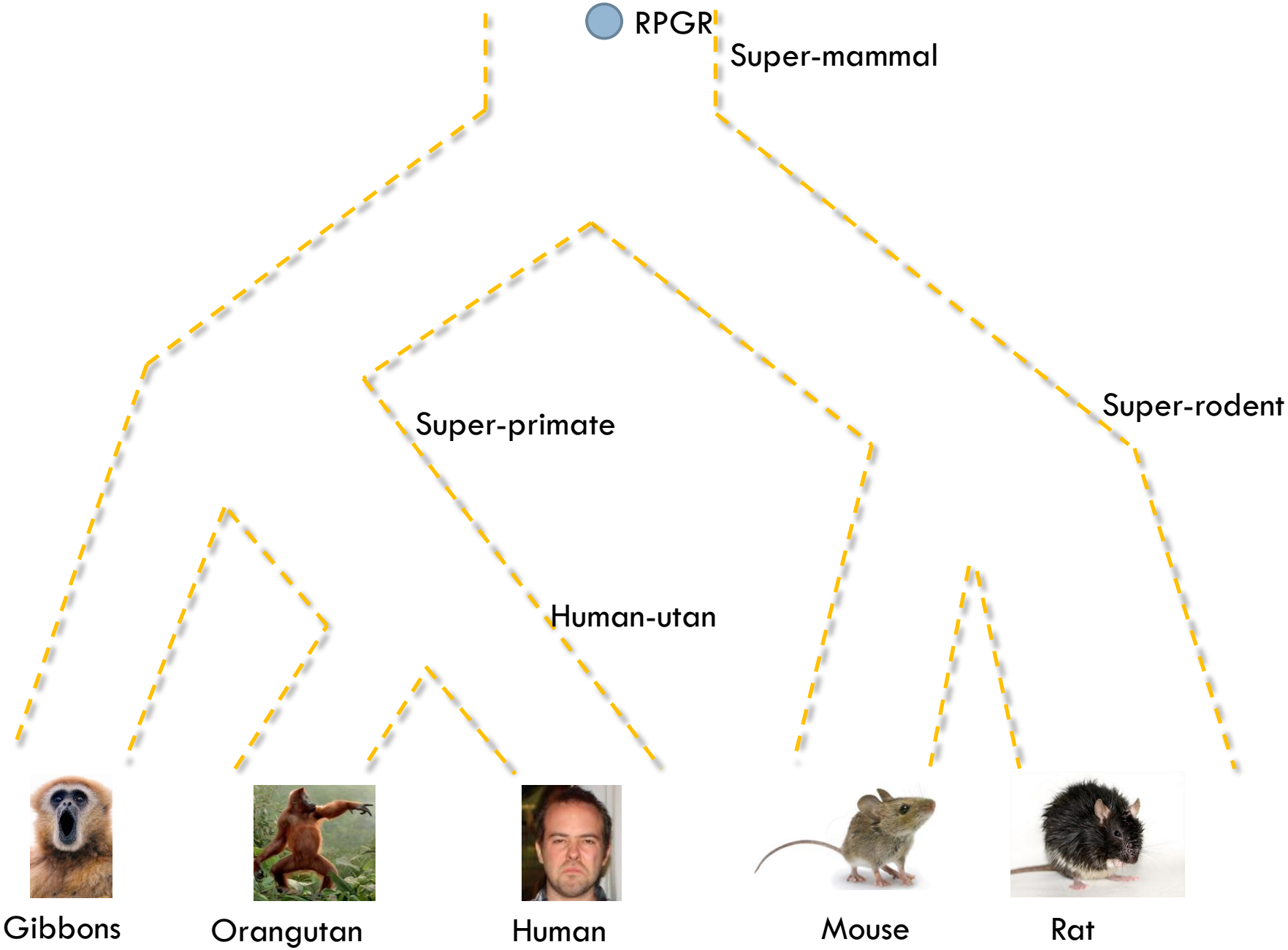
Reconciliation in phylogenomics

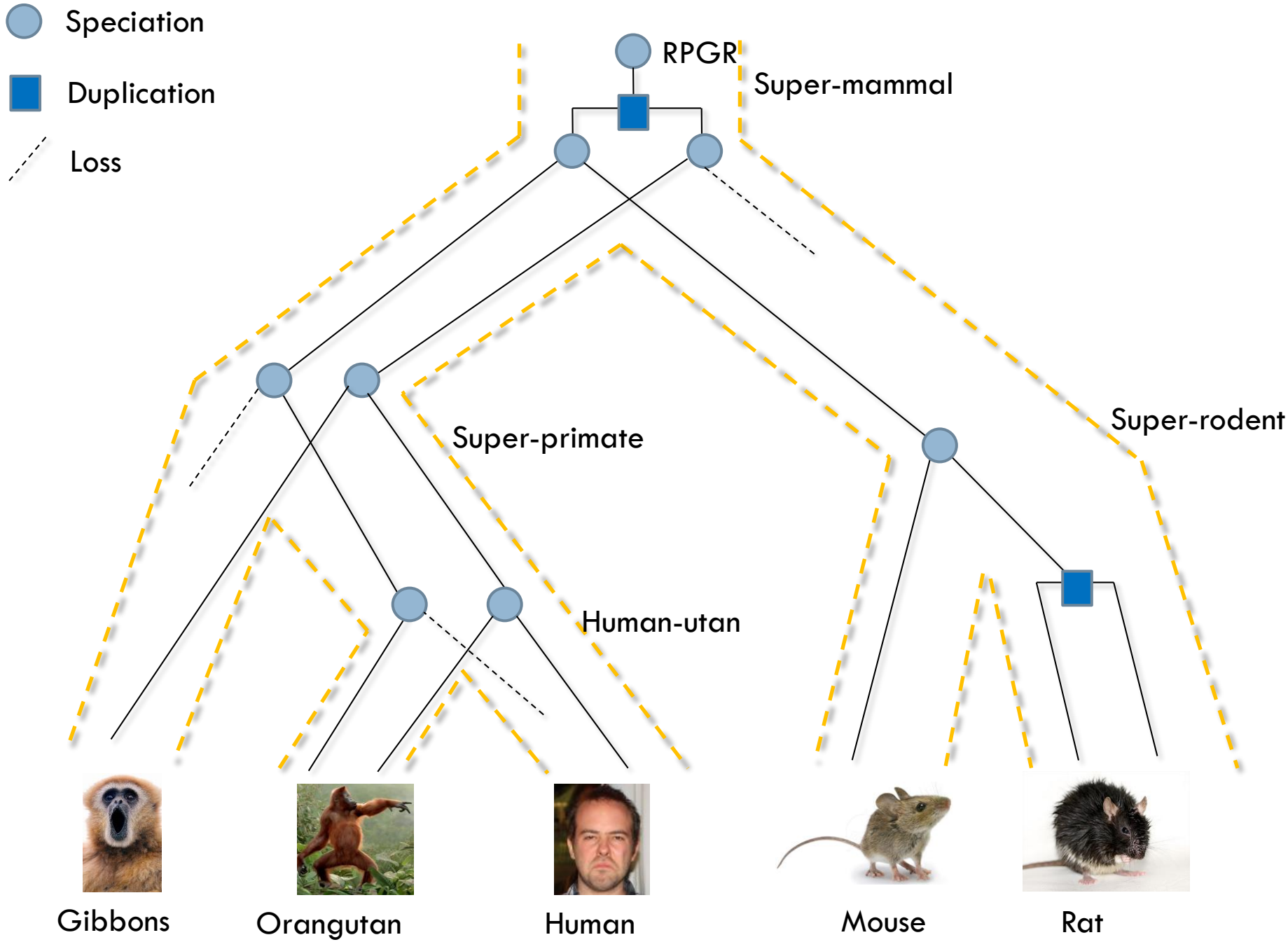
- **Phylogenomics** : evolutionary analysis that involves whole genomes or large portions of it.
- **Traditional reconciliation** : single gene families
- **Phylogenomics reconciliation** : many gene families

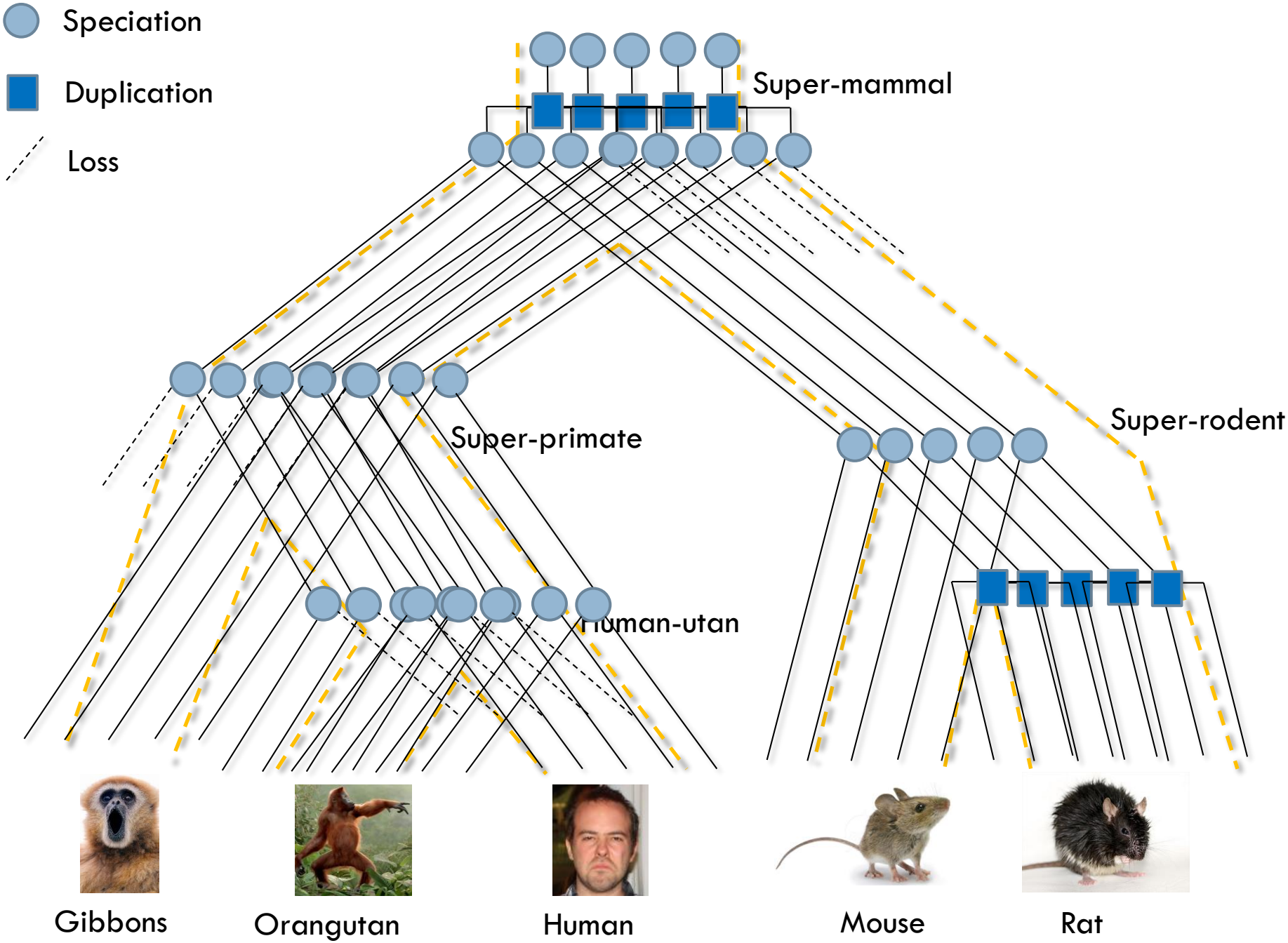
The plan

- Part 1 : basics of multi-gene family reconciliation
- Part 2 : reconciliation with segmental duplications + losses
- Part 3 : reconciling syntenic blocks


Super-mammal
Super-primate
Super-rodent
Human-utan
Gibbons
Orangutan
Human
Mouse
Rat


RPGR
Super-mammal
Super-primate
Super-rodent
Human-utan
Gibbons
Orangutan
Human
Mouse
Rat


Speciation
Duplication
Loss
RPGR
Super-mammal
Super-primate
Super-rodent
Human-utan
Gibbons
Orangutan
Human
Mouse
Rat


Speciation
Duplication
Loss
Super-mammal
Super-primate
Super-rodent
Human-utan
Gibbons
Orangutan
Human
Mouse
Rat

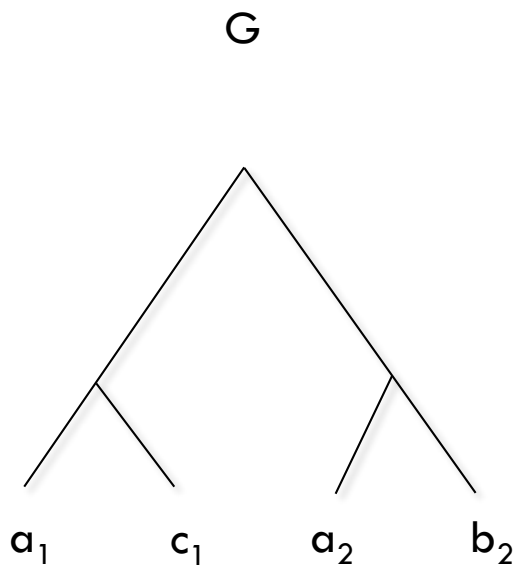
The big question

Given many gene trees, how to identify events that affect genes from several gene trees?

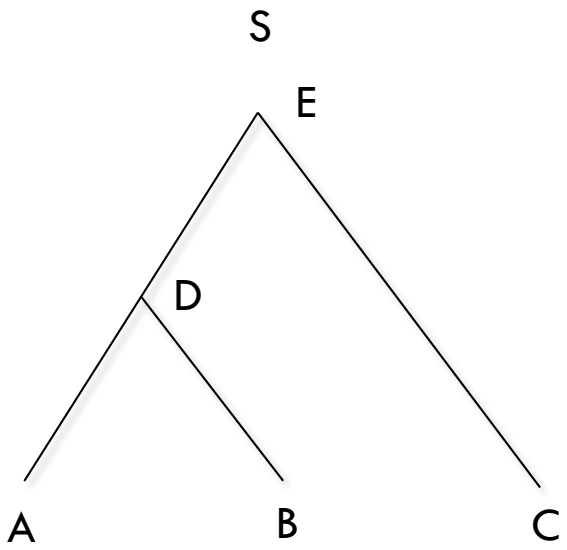
- Segmental duplications, losses, transfers.
- Whole genome duplications followed by block deletions

Reconciliation

Reconciliation identifies **duplication**, **speciation** and **loss** events in a gene tree G, using a species tree S.



Gene tree

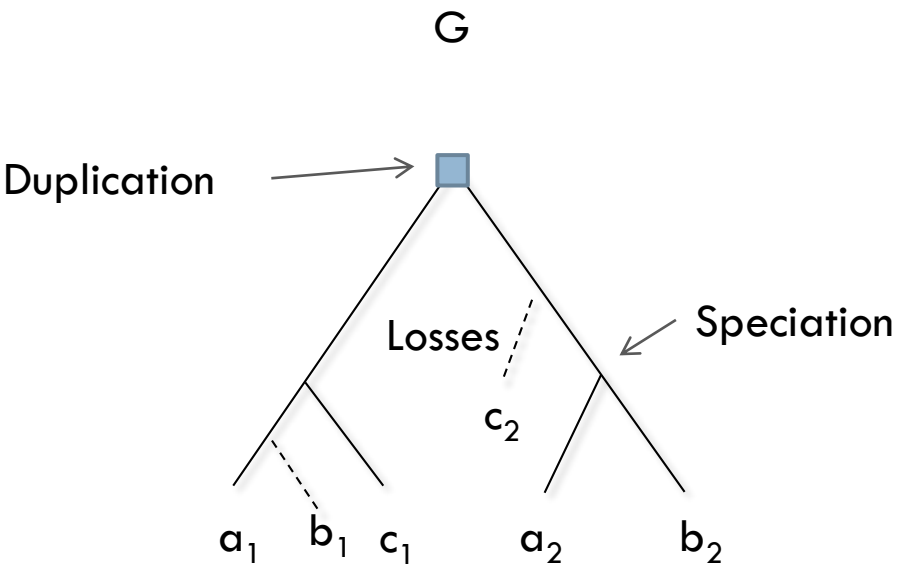


Species tree

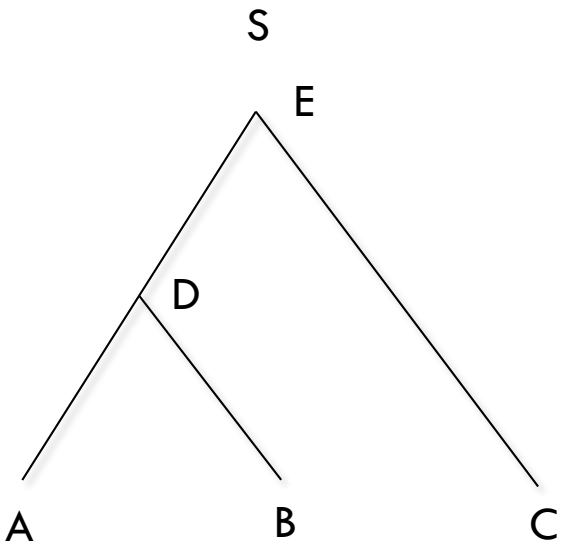
Notation tip: gene name = lowercase species

Reconciliation

Reconciliation identifies **duplication**, **speciation** and **loss** events in a gene tree G, using a species tree S.



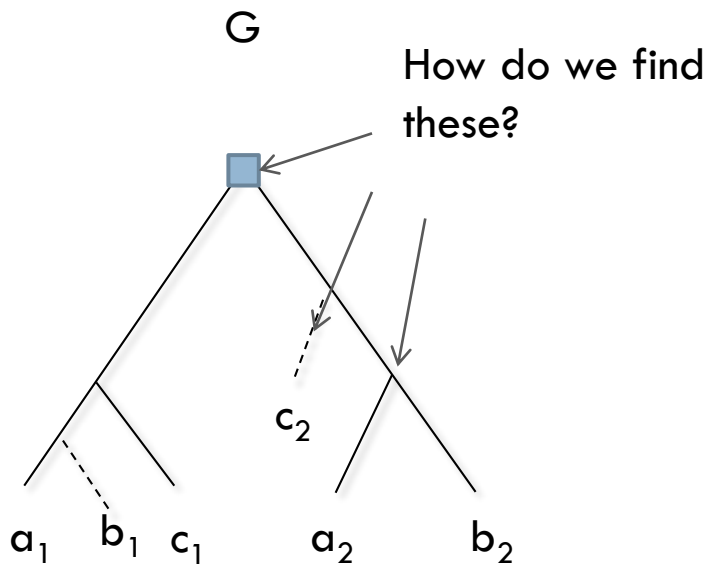
Gene tree



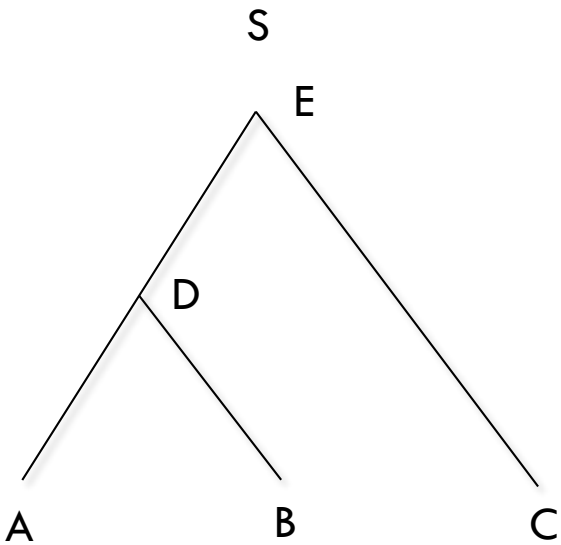
Species tree

Reconciliation

Reconciliation identifies **duplication**, **speciation** and **loss** events in a gene tree G, using a species tree S..

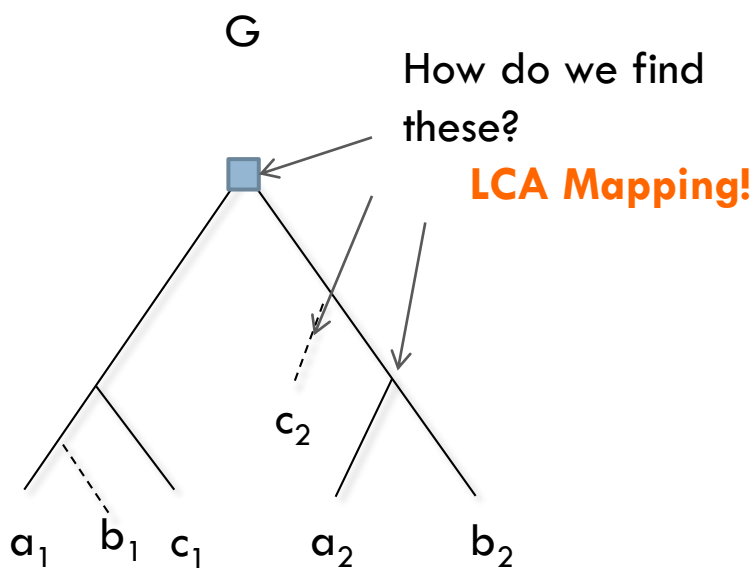


Gene tree

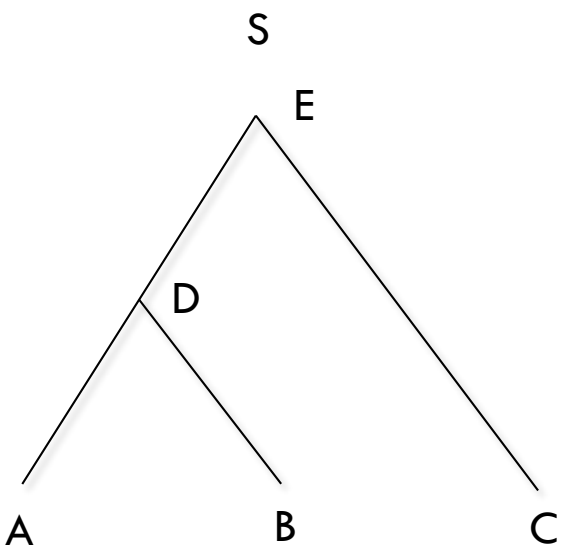


Species tree

LCA Mapping



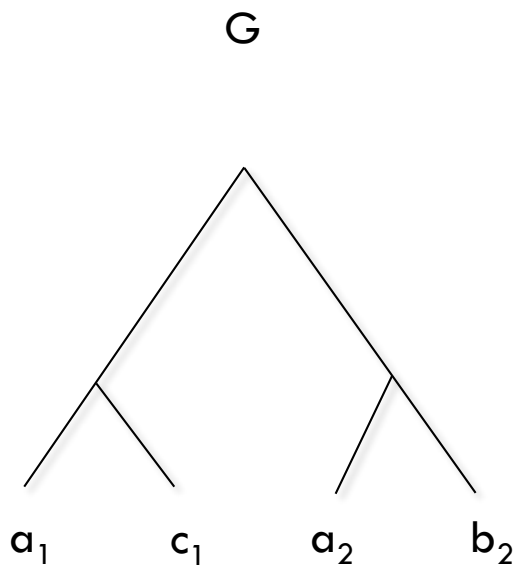
Gene tree



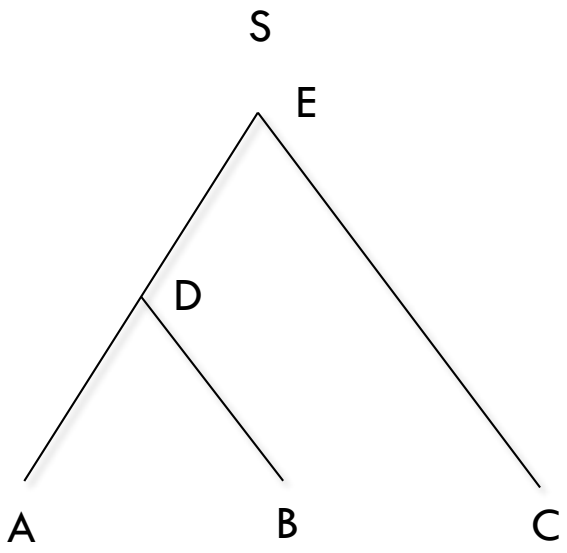
Species tree

LCA Mapping

Map each ancestral gene to the **species** that is the **lowest common ancestor (LCA)** of the descending mapped species.



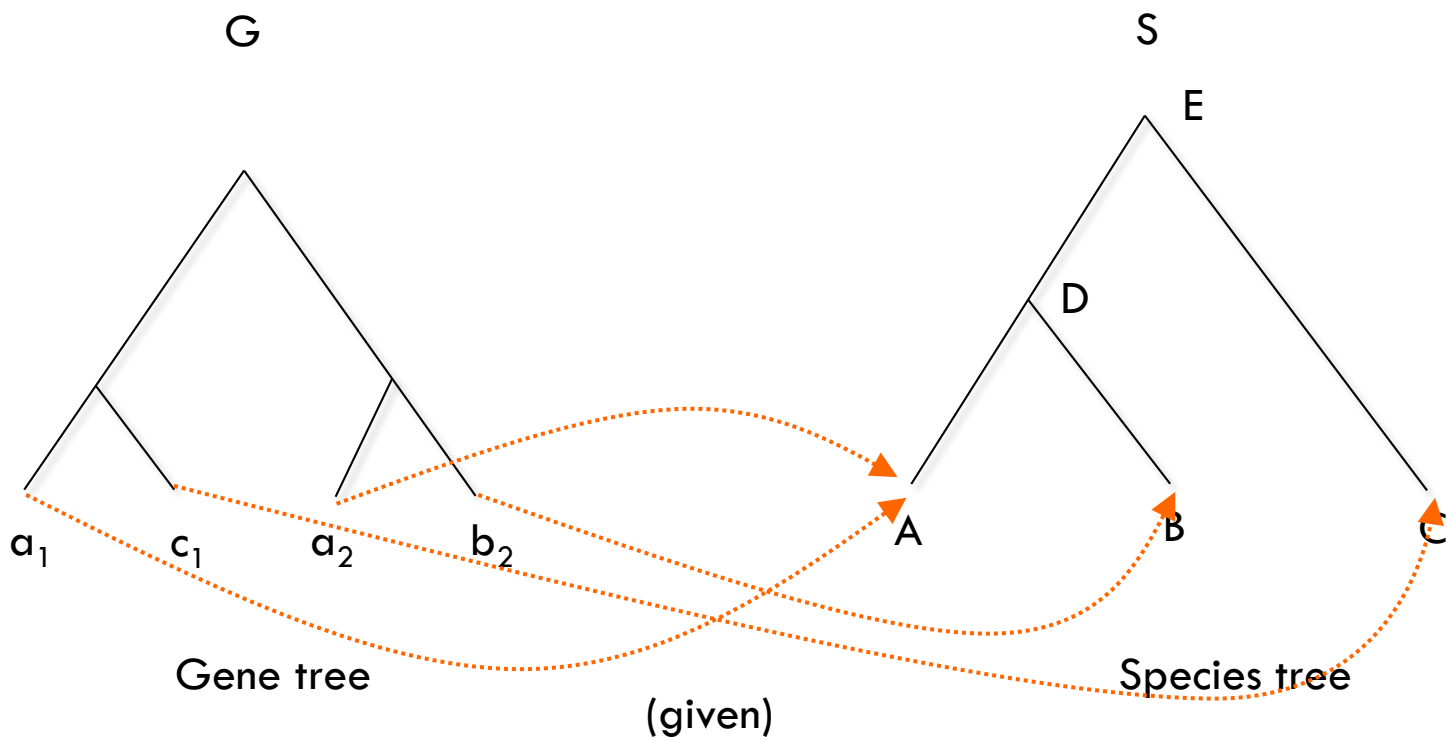
Gene tree



Species tree

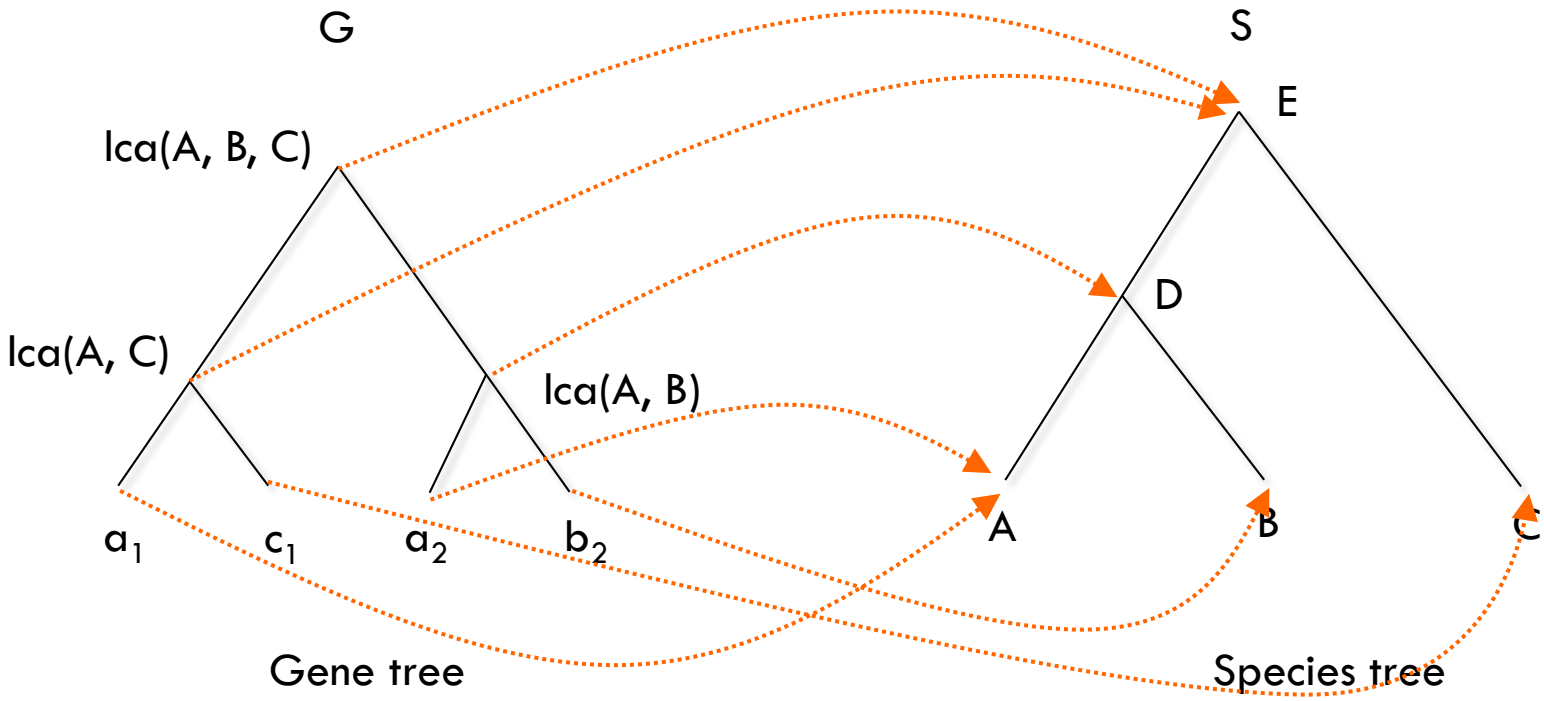
LCA Mapping

Map each ancestral gene to the **species** that is the **lowest common ancestor (LCA)** of the descending mapped species.



LCA Mapping

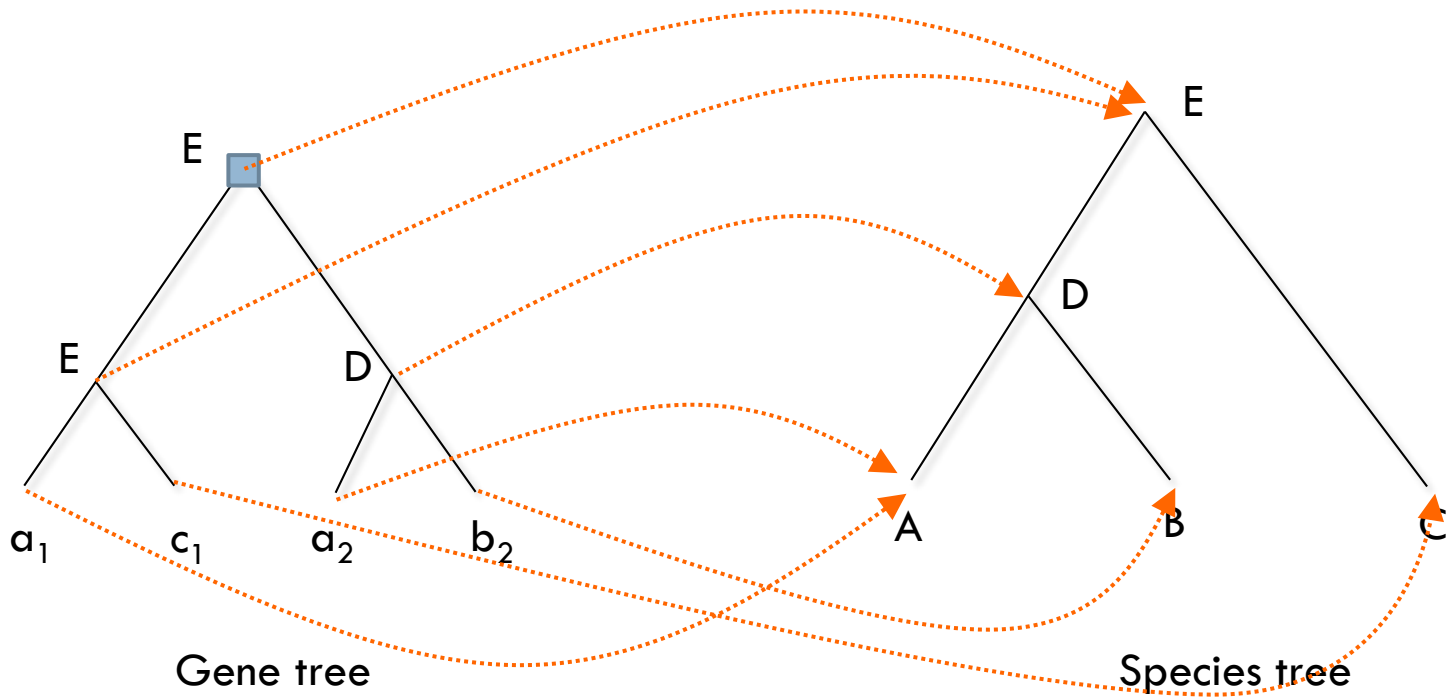
Map each ancestral gene to the **species** that is the **lowest common ancestor (LCA)** of the descending mapped species.



LCA Mapping

Map each ancestral gene to the **species** that is the **lowest common ancestor (LCA)** of the descending mapped species.

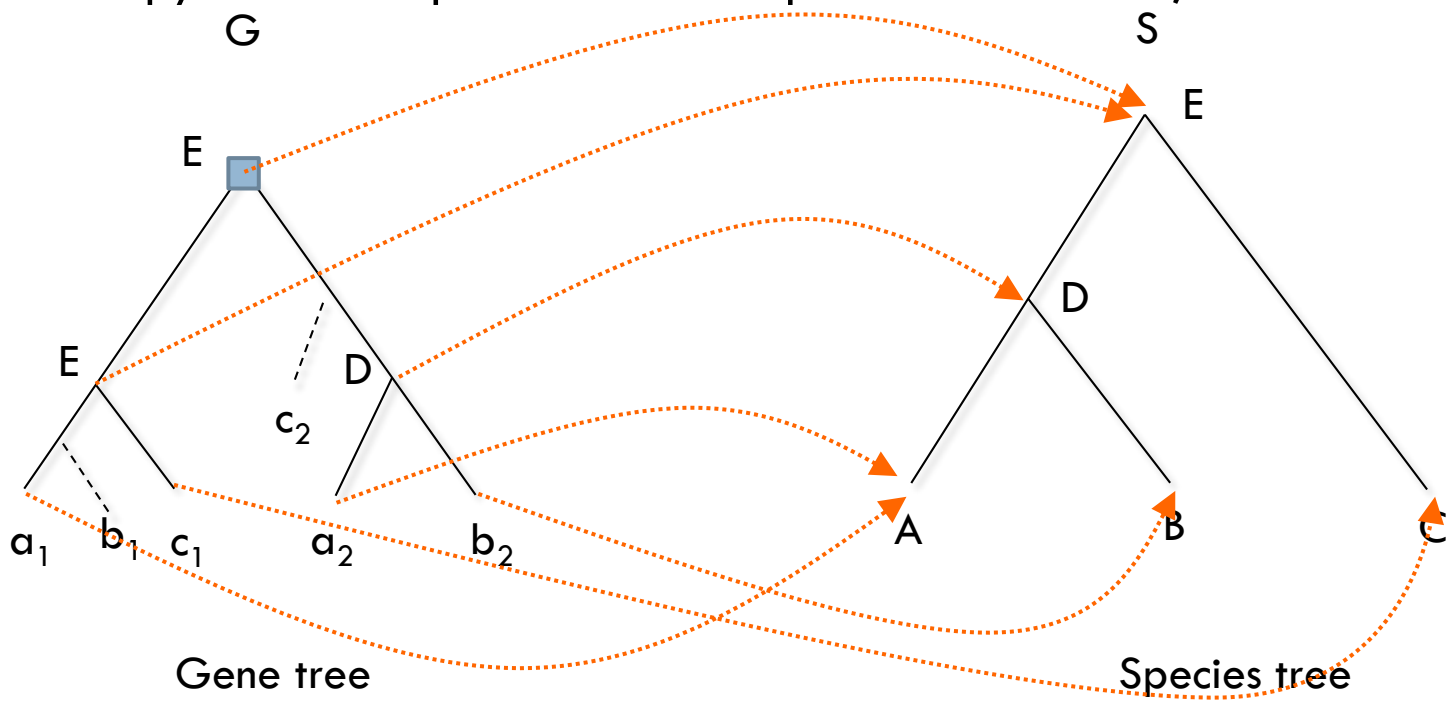
- **Rule**: a node of G **must be a Dup** if it maps to the same species as a child.



LCA Mapping

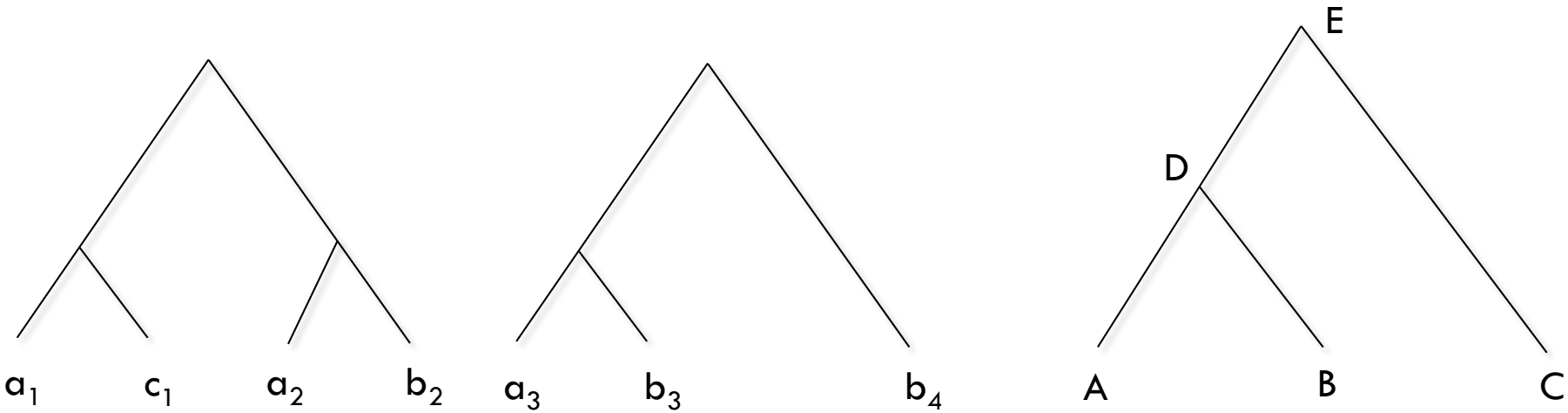
Map each ancestral gene to the **species** that is the **lowest common ancestor (LCA)** of the descending mapped species.

- **Rule**: a node of G **must be a Dup** if it maps to the same species as a child.
- Each copy should be present in each species – otherwise, losses.



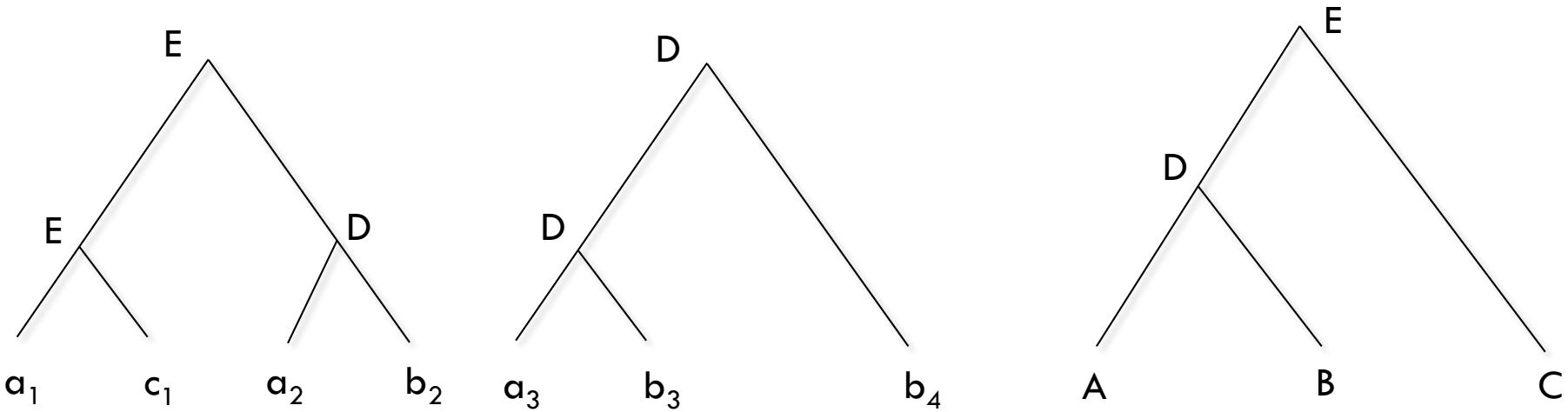
LCA Mapping

Now let's have more than one gene tree.



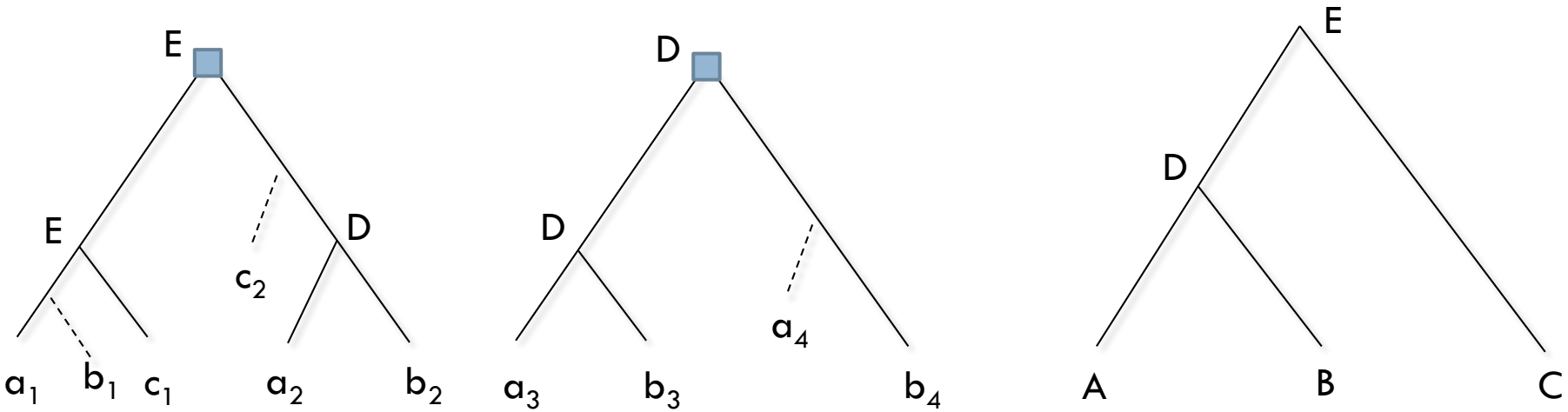
LCA Mapping

Now let's have more than one gene tree.



LCA Mapping

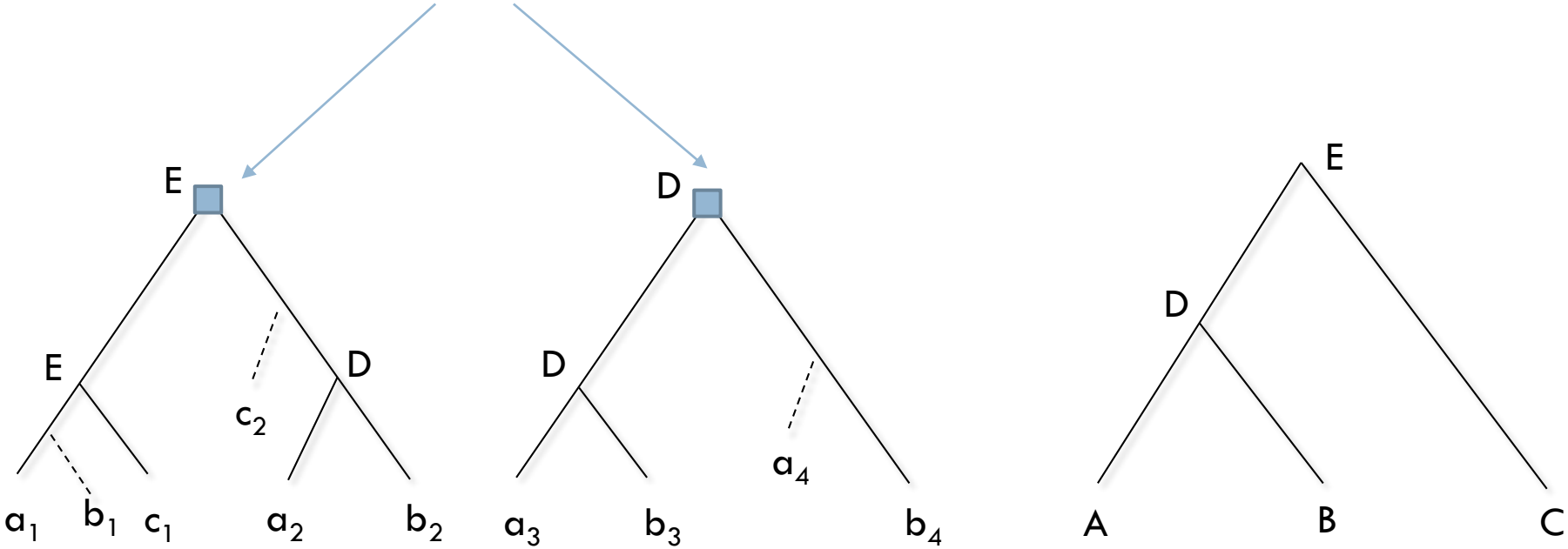
Now let's have more than one gene tree.



LCA Mapping

Now let's have more than one gene tree.

Maybe these duplications **are the same**! (e.g. a block duplication of a segment)



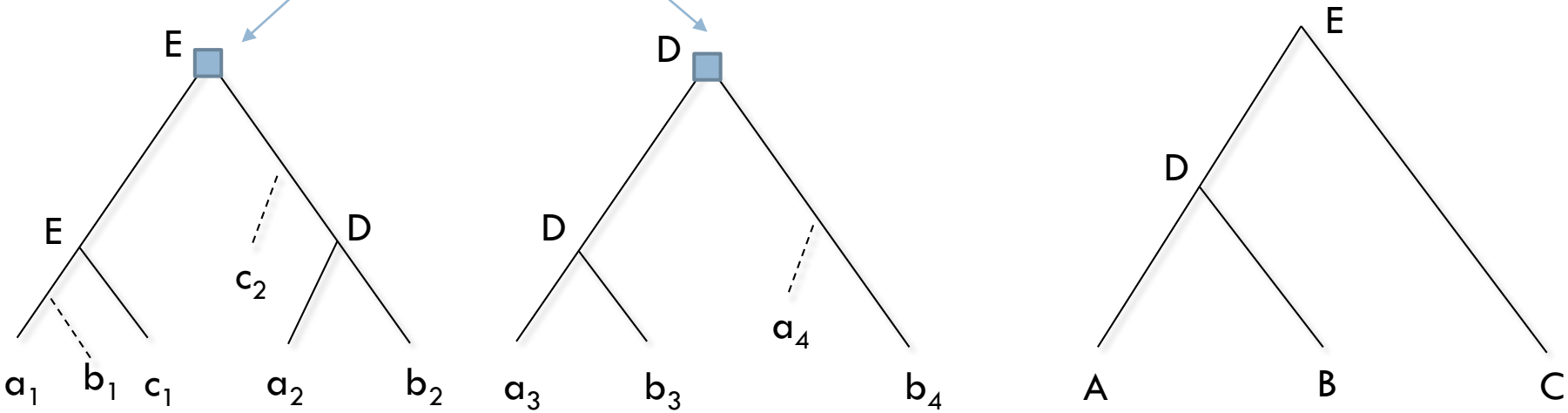
LCA Mapping

Now let's have more than one gene tree.

Maybe these duplications **are the same**! (e.g. a block duplication of a segment)

If so, this Dup must have occurred in the E species.

=> We must **remap** the D duplication.



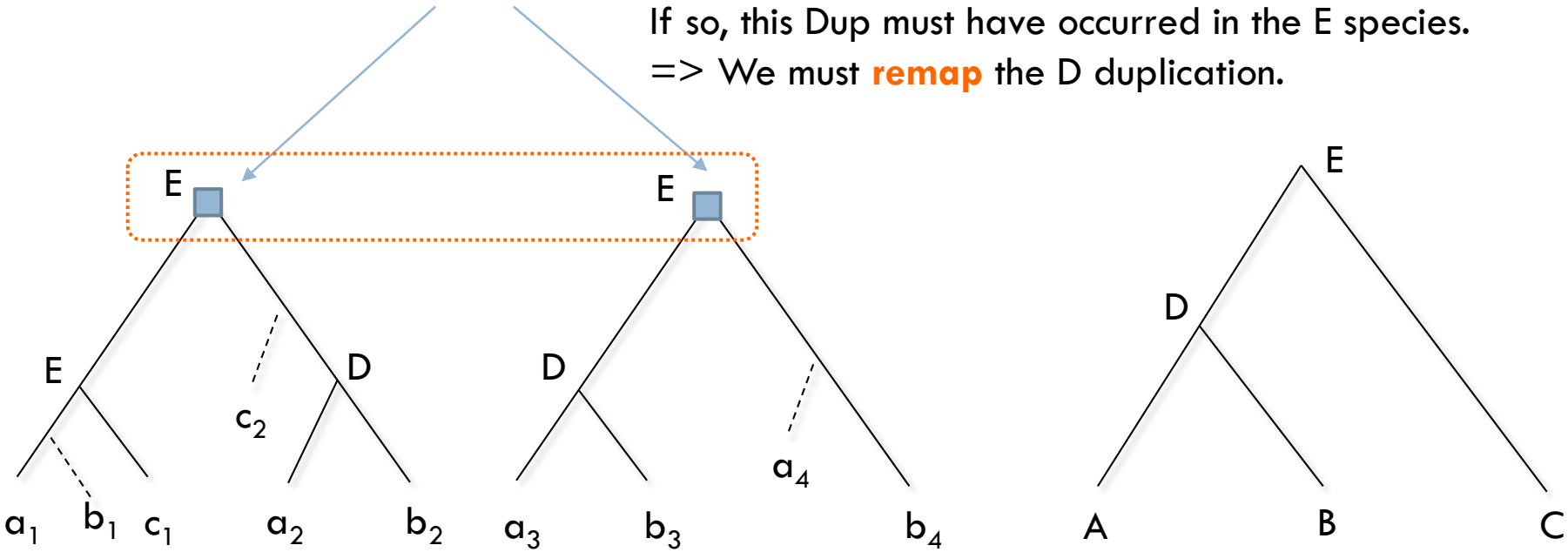
LCA Mapping

Now let's have more than one gene tree.

Maybe these duplications **are the same**! (e.g. a block duplication of a segment)

If so, this Dup must have occurred in the E species.

=> We must **remap** the D duplication.



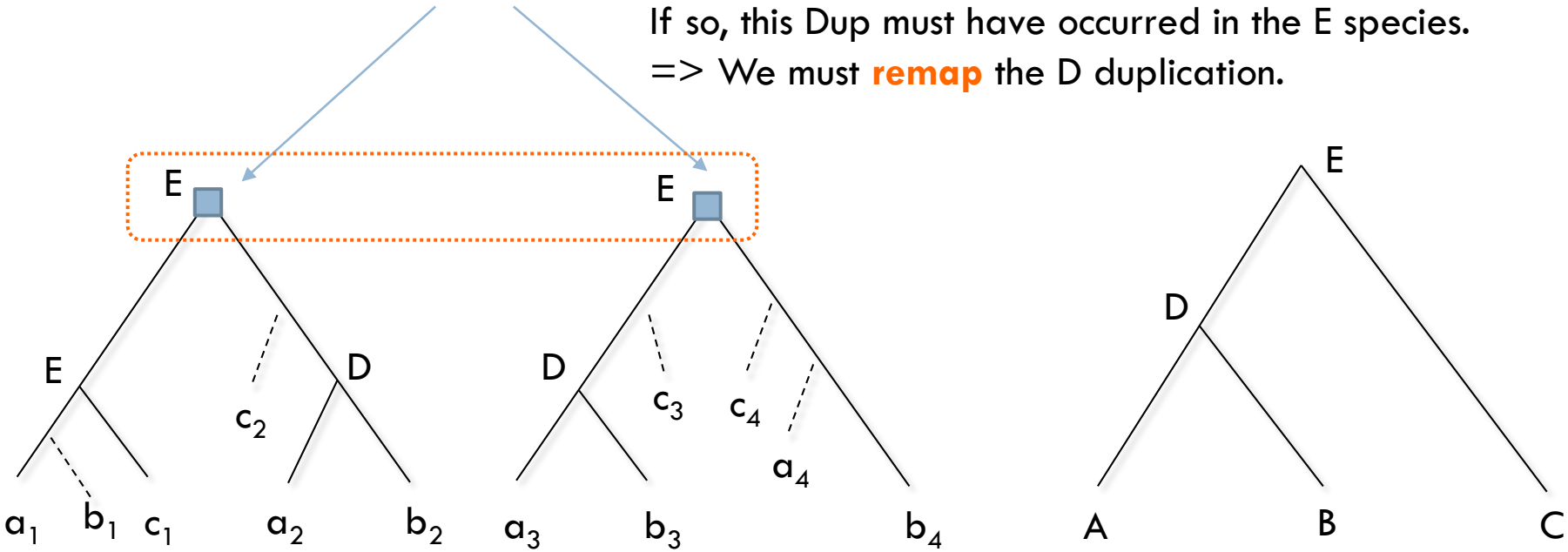
LCA Mapping

Now let's have more than one gene tree.

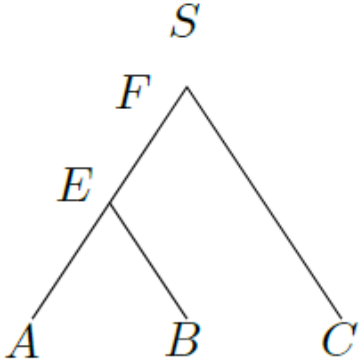
Maybe these duplications **are the same**! (e.g. a block duplication of a segment)

If so, this Dup must have occurred in the E species.

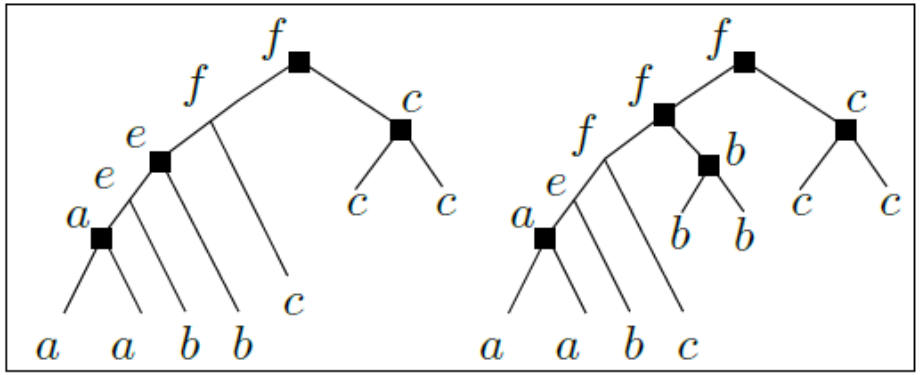
=> We must **remap** the D duplication.



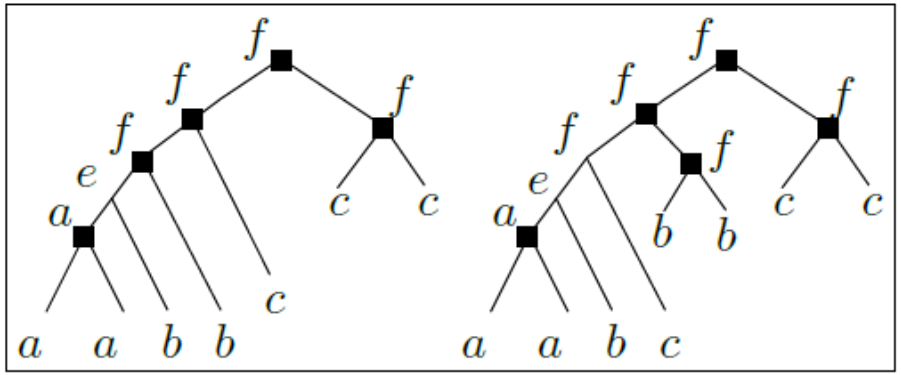
1 DUP, 5 LOSSES (before, we had 2 DUPS, 3 LOSSES)



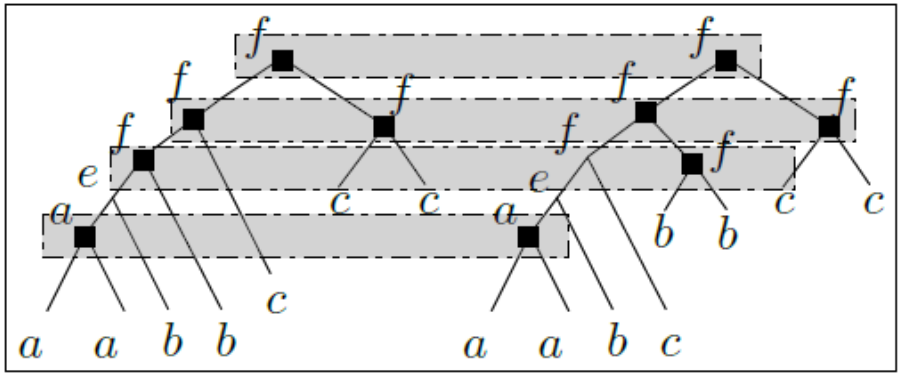
(1)



1) Classical lca mapping



2) Remap



3) Find dups

Axioms of gene-species maps

- A map $m : V(G) \quad V(S)$ is valid if
  - For a leaf u, $m(u)$ is the known species of gene u
  - **Time-consistency** : $m(u) \preccurlyeq m(parent(u))$ for all non-root u.

- A node u of G is a Dup if either
  - $m(u) = m(u')$ for some child u' of u; or
  - $m(u) \neq lca - map(u)$

Axioms of gene-species maps

- A map $m : V(G) \quad V(S)$ is valid if
  - For a leaf u, $m(u)$ is the known species of gene u
  - **Time-consistency** : $m(u) \preccurlyeq m(parent(u))$ for all non-root u.

- A node u of G is a Dup if either
  - $m(u) = m(u')$ for some child u' of u; or
  - $m(u) \neq lca - map(u)$

- A gene loss must be inferred on the uv branch for each species strictly between $m(u)$ and $m(v)$
  - But including $m(u)$ if u is a Dup

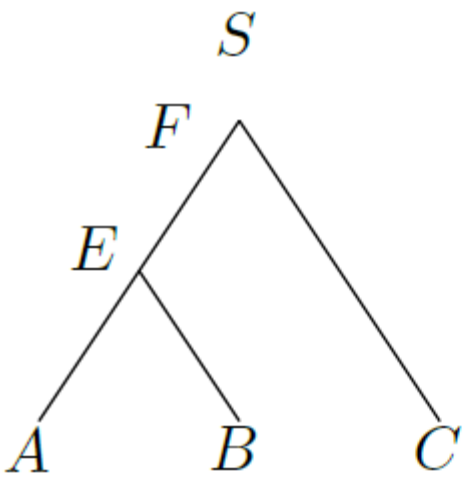
A brief survey of models of segmental duplications

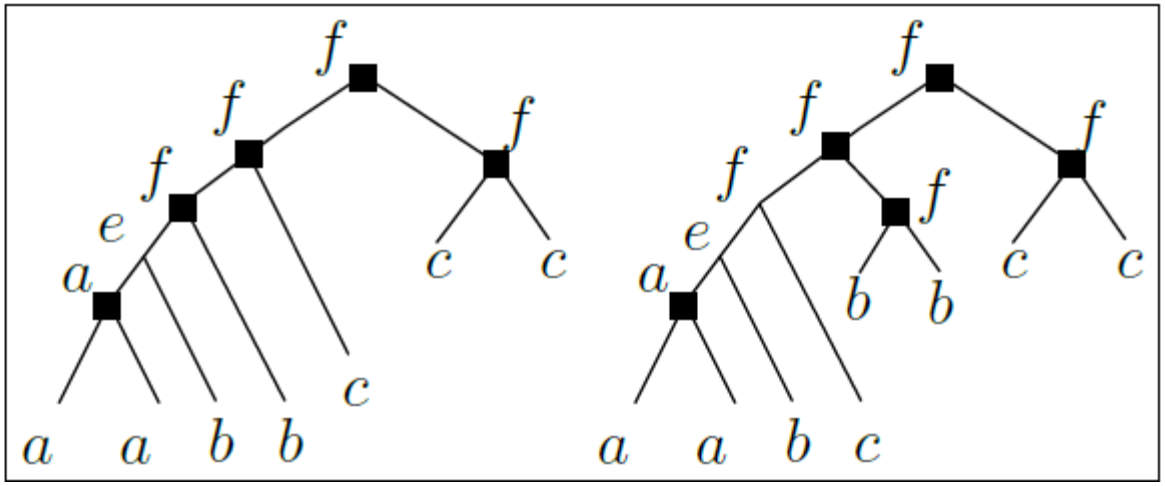
Models of segmental duplications

- Episode clustering EC
  - *[Guigo, Muchnik & Smith, Mol. Phylo & evol 1996]*
  - Dup events contain **all genes in the same species**.
- Gene duplication clustering GD
  - *[Fellows, Hallett & Stege, ISAAC 1998]*
  - Dup events have **at most 1 gene per gene tree**.
- Minimum episode ME
  - *[Bansal & Eulenstein, Bioinformatics 2008]*
  - Dup events **do not contain a gene and one of its descendants**.

Models of segmental duplications

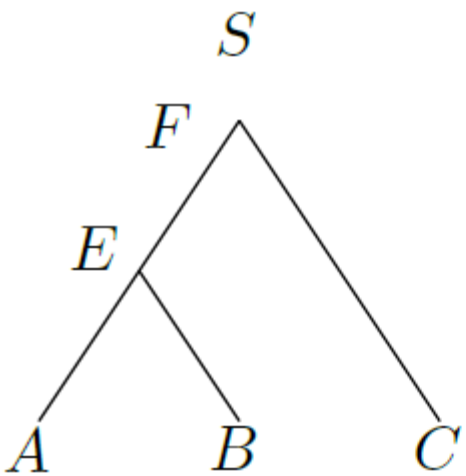
- Episode clustering EC *[Guigo 1996]*
  - Dup events can affect **all** genes in the same species.



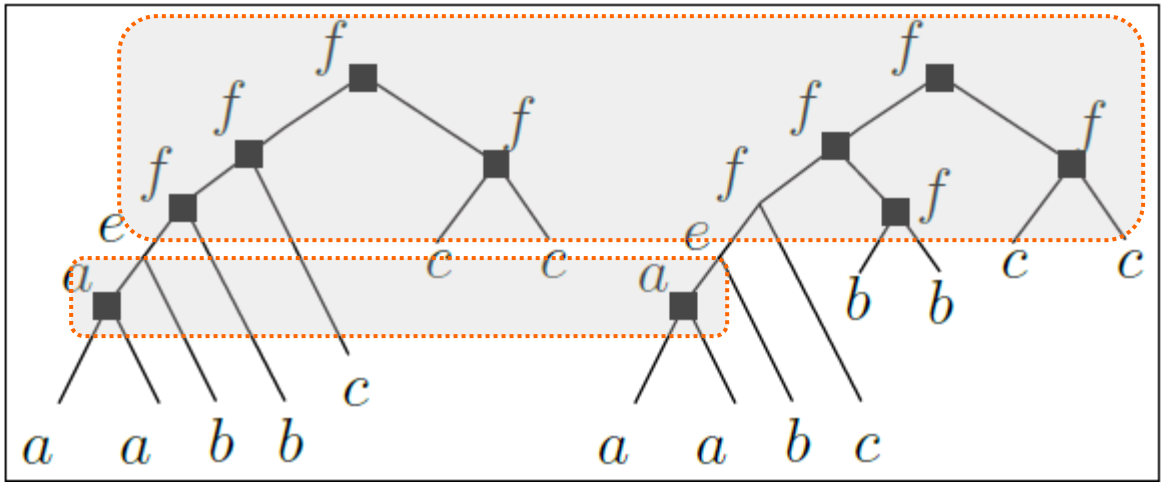


Models of segmental duplications

- Episode clustering EC
  - Dup events can affect **all** genes in the same species.

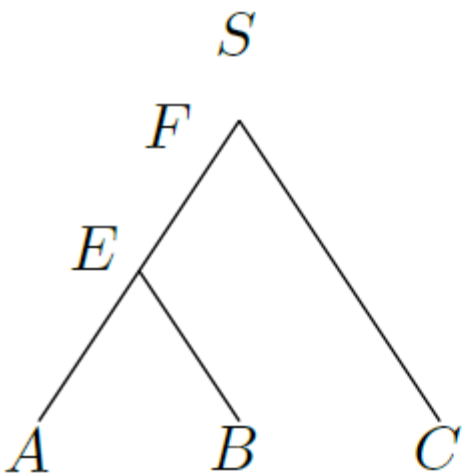


# of dups = 2

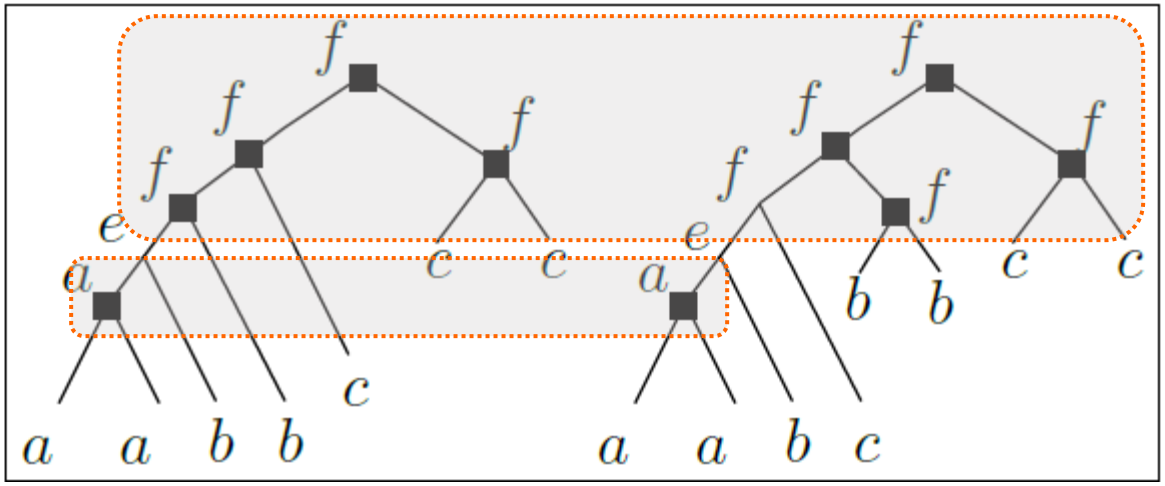


Models of segmental duplications

- Episode clustering EC
  - Dup events can affect **all** genes in the same species.
  - Why not just remap every gene to F and have a single dup? Restriction needed.

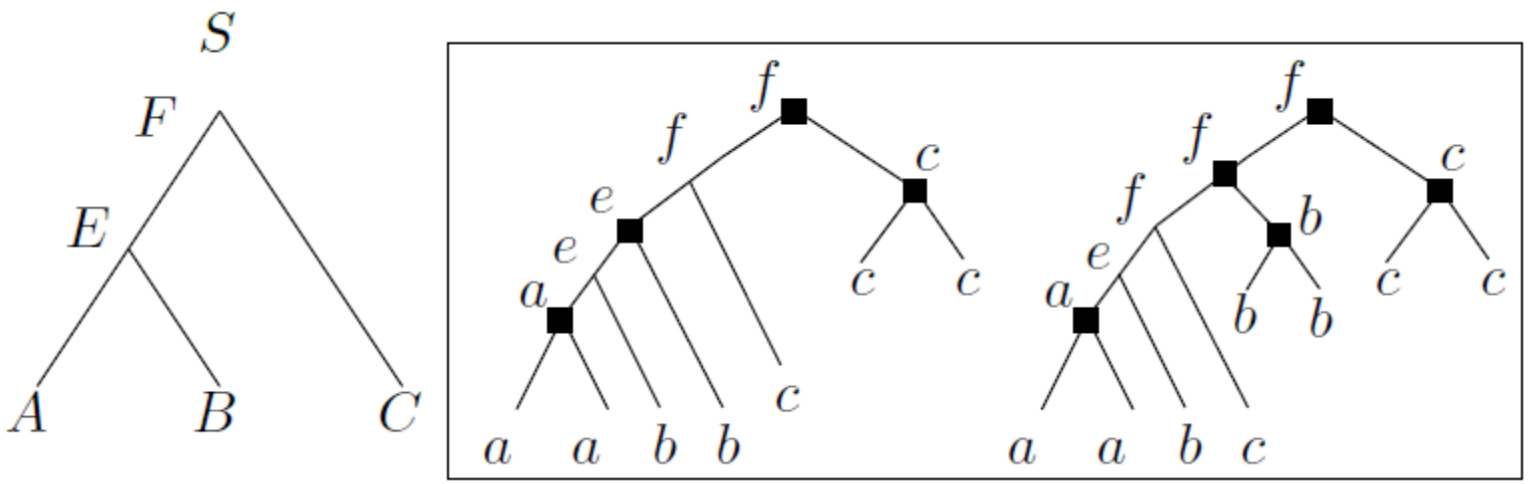


# of dups = 2



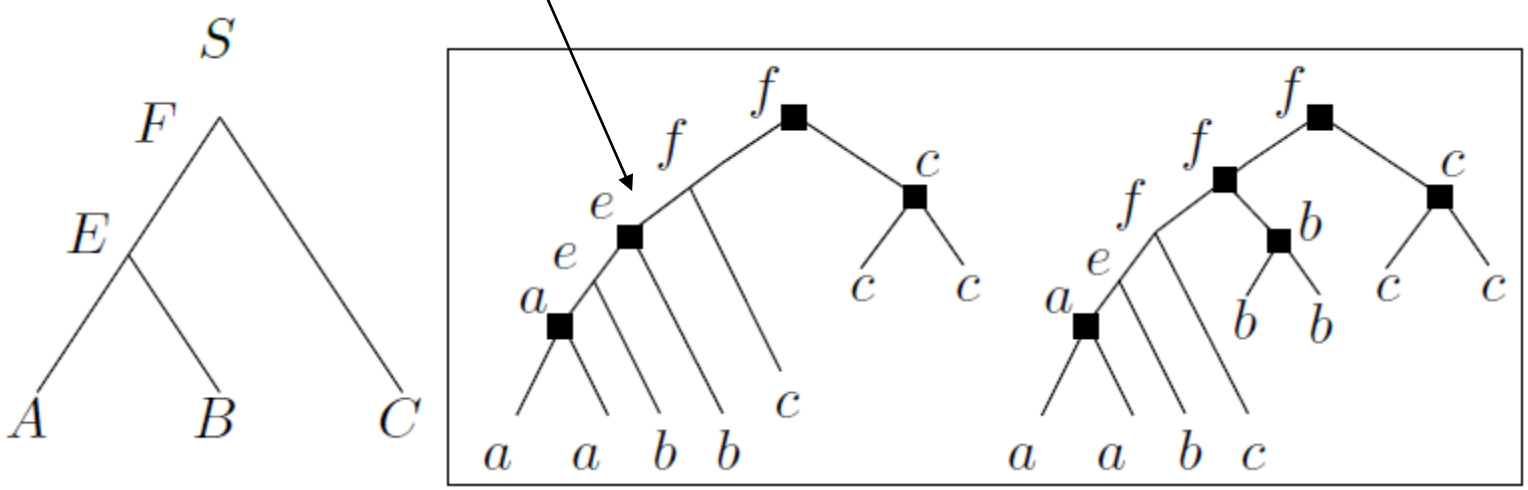
Models of segmental duplications

- Episode clustering EC
  - Restriction : if a node can be a speciation, it shall remain a speciation.



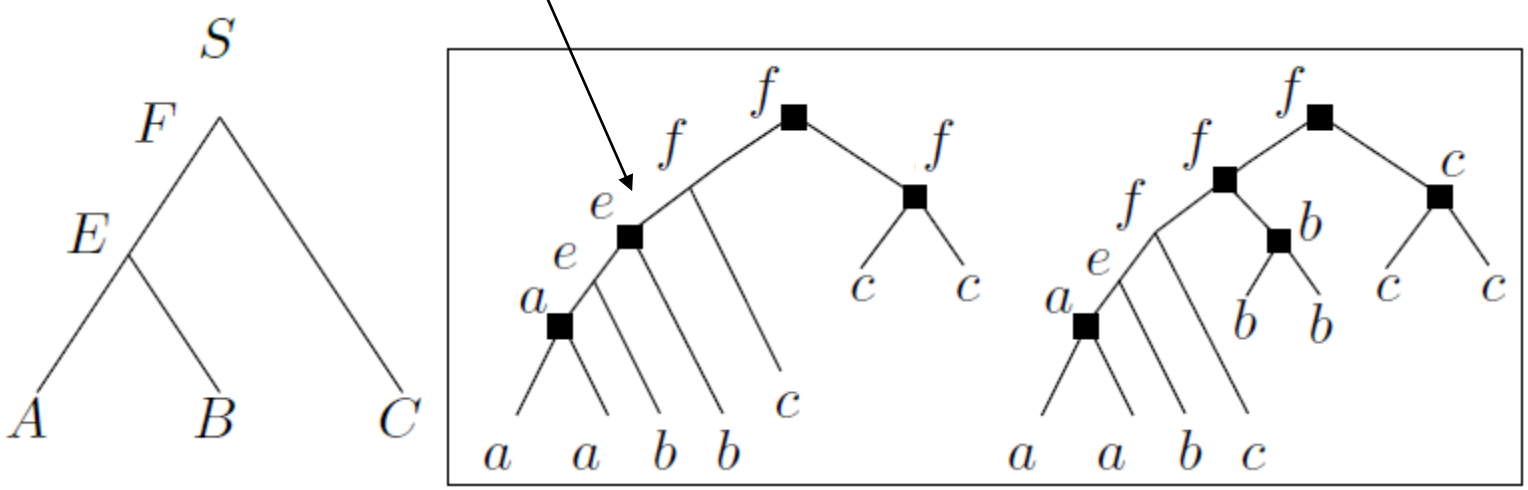
Models of segmental duplications

- Episode clustering EC
  - Restriction : if a node can be a speciation, it shall remain a speciation.
  - Remapping e to f would prevent the speciation => forbidden.



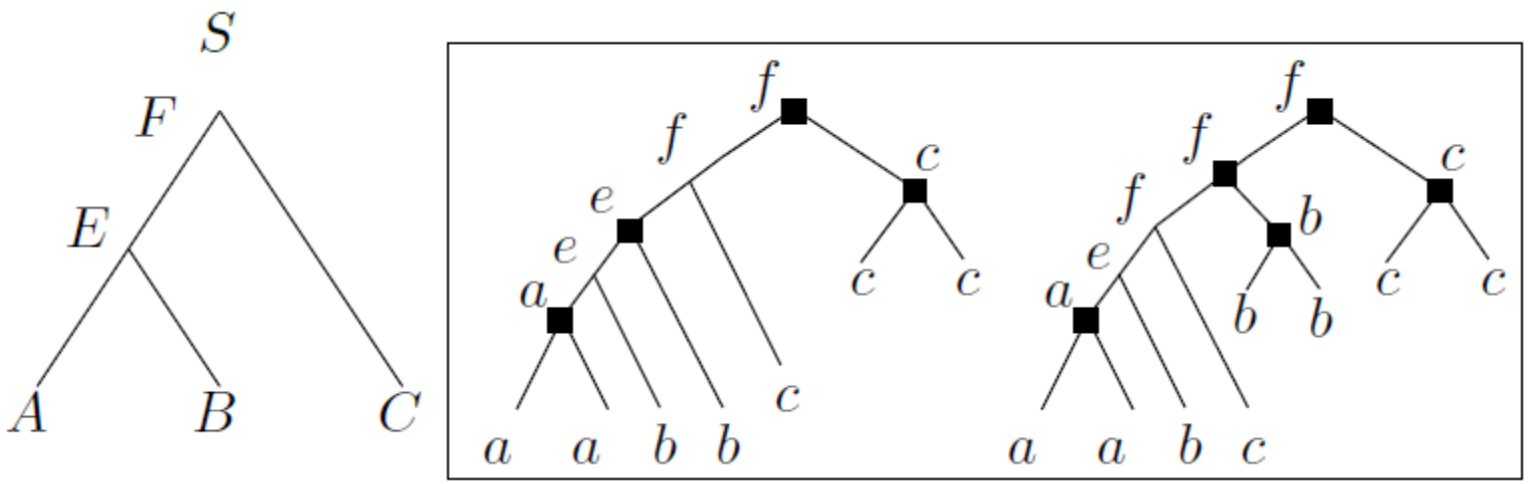
Models of segmental duplications

- Episode clustering EC
  - Restriction : if a node can be a speciation, it shall remain a speciation.
  - Remapping e to f would prevent the speciation => forbidden.



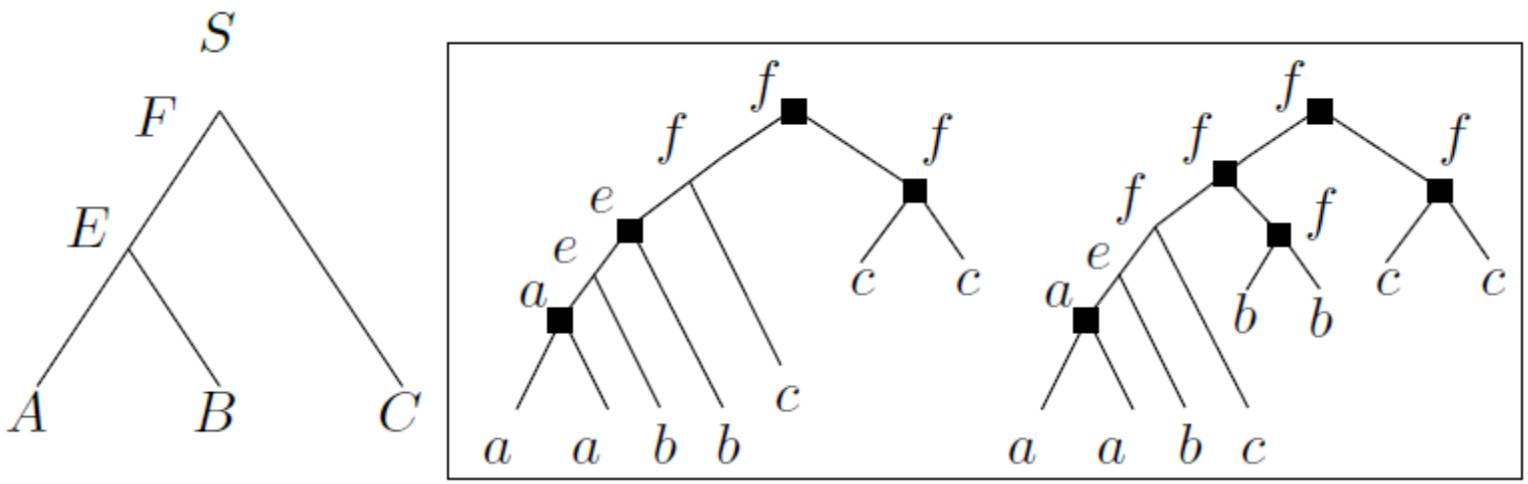
Models of segmental duplications

- Episode clustering EC
  - Goal : find a valid map of the genes that does not break any speciation and minimizes # of species that have at least one Dup in them.



Models of segmental duplications

- Episode clustering EC
  - Goal : find a valid map of the genes that does not break any speciation and minimizes # of species that have at least one Dup in them.



Models of segmental duplications

- Episode clustering EC
  - Goal : find a valid map of the genes that does not break any speciation and minimizes # of species that have at least one Dup in them.

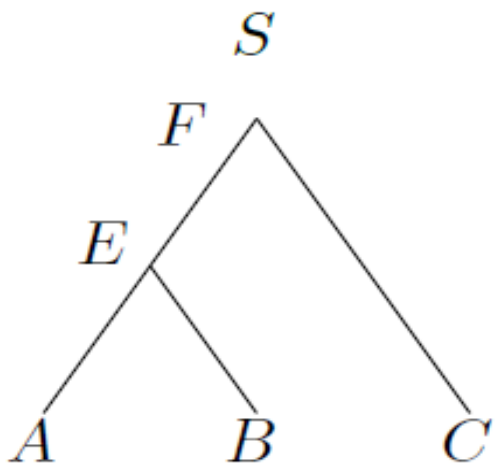
- Can be solved in polynomial time (also for other types of restrictions).
  - *[Burleigh & al., RECOMB 2008]*

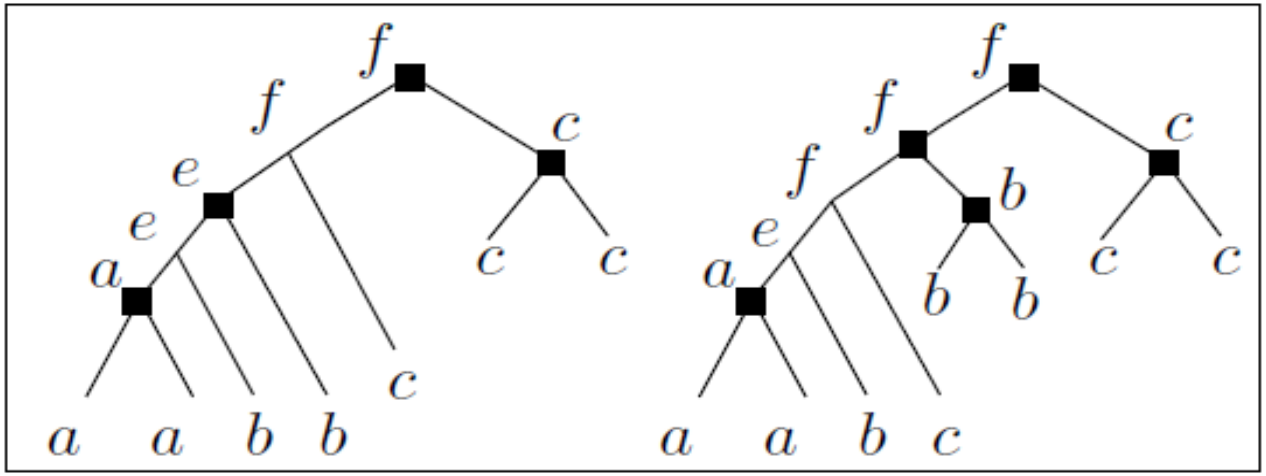
Models of segmental duplications

- Gene duplication clustering GD *[FHS 1998]*
  - Dup events can affect genes in the same species, but contain **at most one gene per gene tree**.
  - No restriction on mapping.

Models of segmental duplications

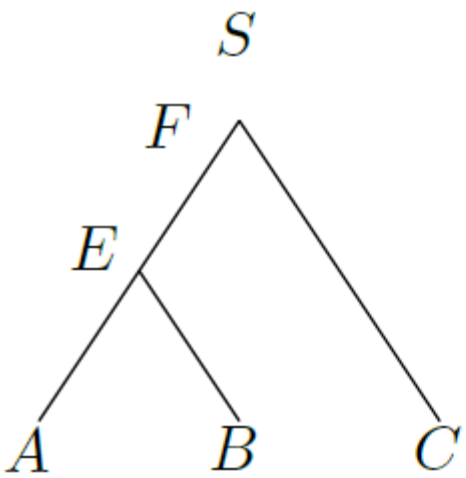
- Gene duplication clustering GD
  - Dup events can affect genes in the same species, but contain **at most one gene per gene tree**.
  - No restriction on mapping.

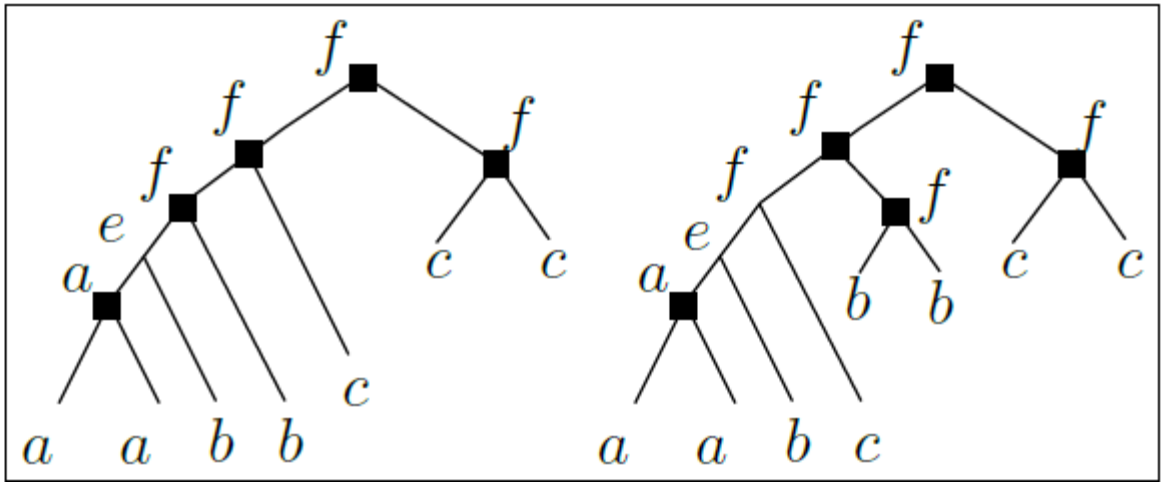




Models of segmental duplications

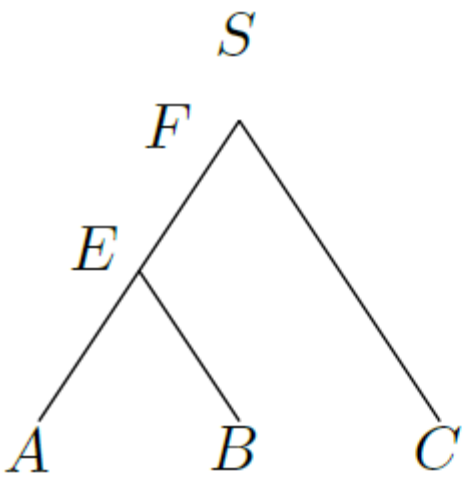
- Gene duplication clustering GD
  - Dup events can affect genes in the same species, but contain **at most one gene per gene tree**.
  - No restriction on mapping.

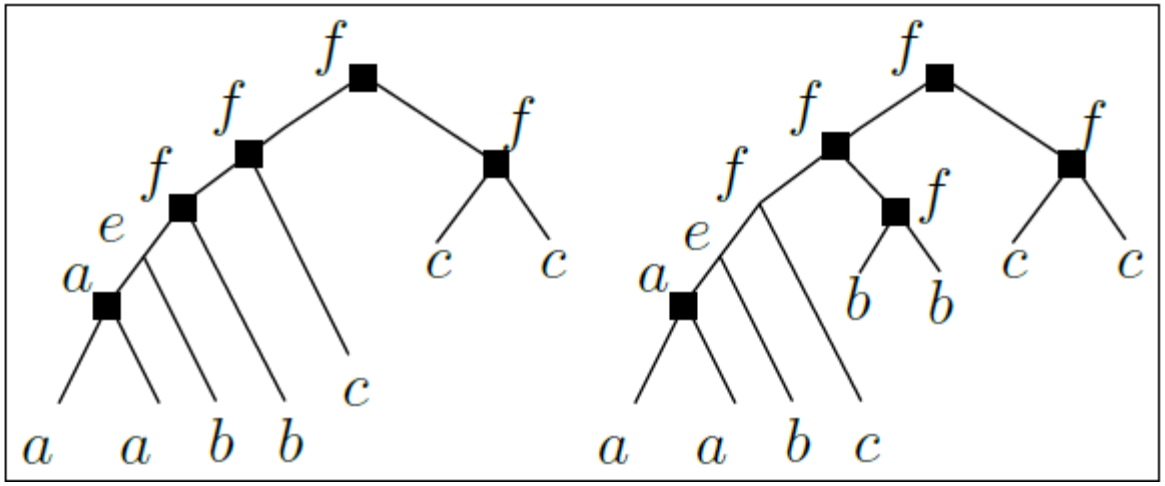




Models of segmental duplications

- Gene duplication clustering GD
  - Dup events can affect genes in the same species, but contain **at most one gene per gene tree**.
  - No restriction on mapping.





Models of segmental duplications

- Gene duplication clustering GD
  - Dup events can affect genes in the same species, but contain **at most one gene per gene tree**.
  - No restriction on mapping.

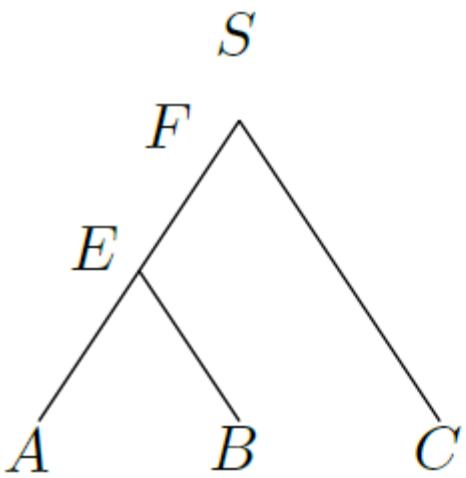
- NP-hard, and even W[1]-hard in the # of dups.
  - *[FHS, ISAAC 2008]*

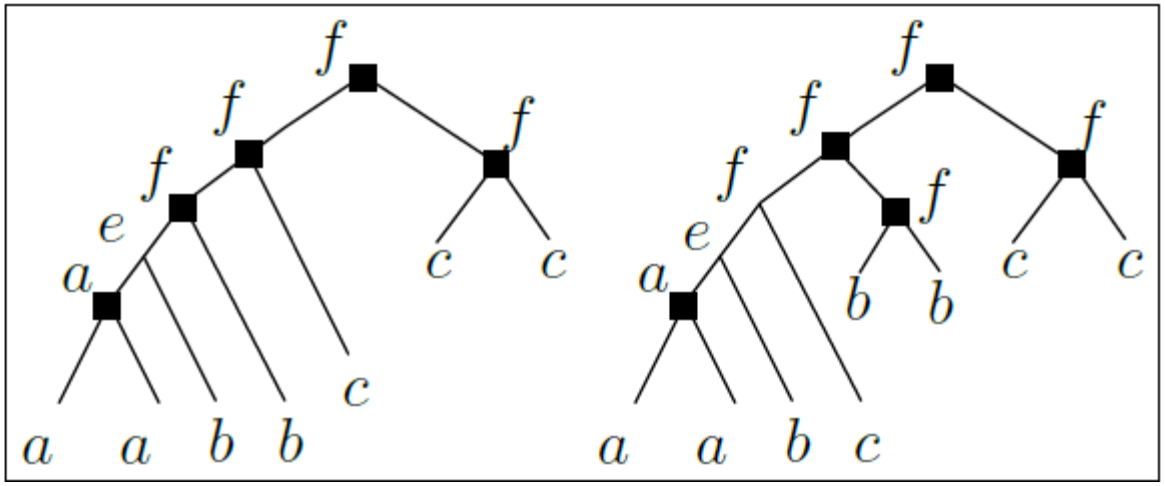
Models of segmental duplications

- Minimum episode ME *[Bansal & Eulenstein 2008]*
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.
  - No remapping restriction needed.

Models of segmental duplications

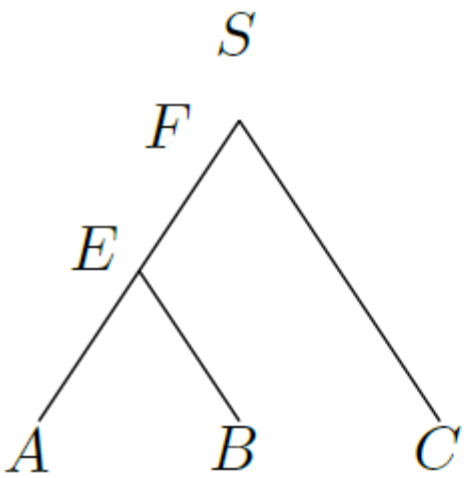
- Minimum episode ME
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.
  - No remapping restriction needed.

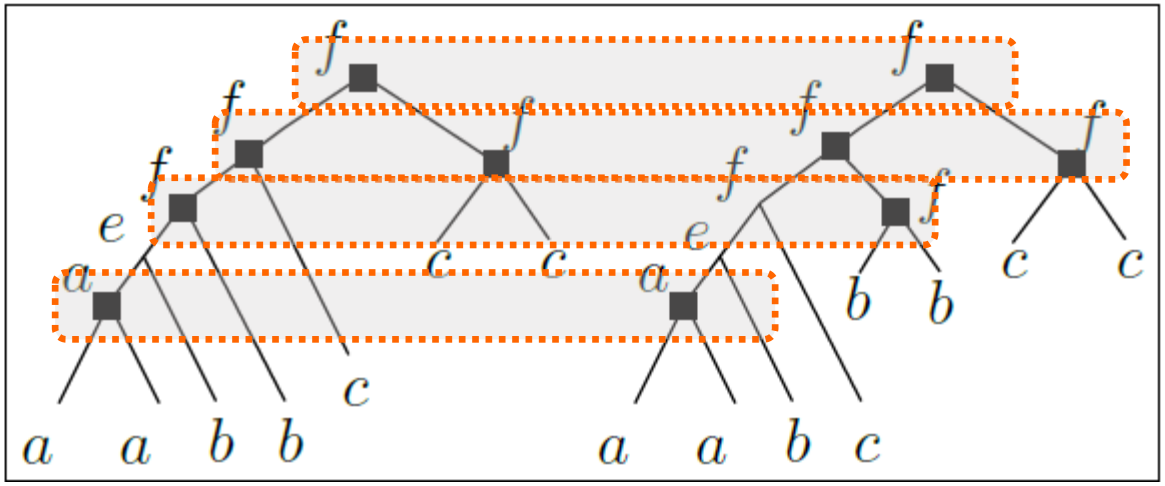




Models of segmental duplications

- Minimum episode ME
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.
  - No remapping restriction needed.





Models of segmental duplications

- Minimum Episodes inference problem
  - **Input** : species tree S, gene trees $G_1, \ldots, G_n$
  - **Find** : a valid gene-species mapping m that minimizes the number of ME duplications.

Models of segmental duplications

- Minimum Episodes inference problem
  - **Input** : species tree S, gene trees $G_1, \ldots, G_n$
  - **Find** : a valid gene-species mapping m that minimizes the number of ME duplications.

- With restriction of "never break a speciation", can be solved in polynomial time.
  - *[Bansal & Eulenstein, Bioinformatics 2008]*
  - *[Paszek & Gorecki, TCBB 2017]*
- Unrestricted mapping = open problem until recently

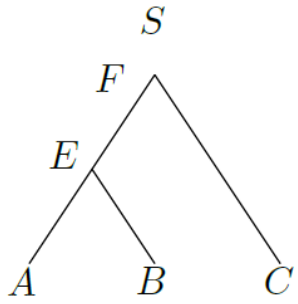
Models of segmental duplications

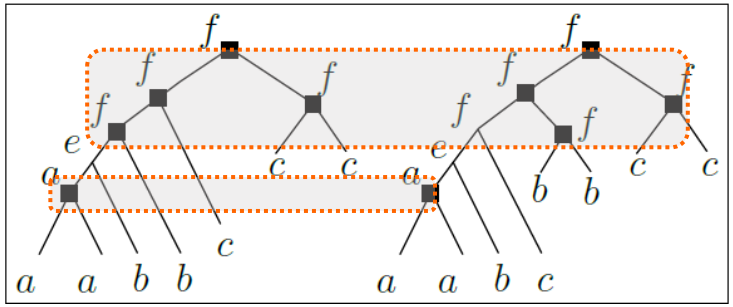
- Minimum Episode and Species Tree Inference
  - **Input** : gene trees $G_1, \ldots, G_n$
  - **Find** : **a species tree S** and a valid gene-species mapping that minimizes the number of ME duplications.

- Can be solved in polynomial time!
  - *[Van Iersel, Janssen, Jones, Murakami & Zeh, TCBB 2019]*
  - Reduction to Beaded Tree problem.

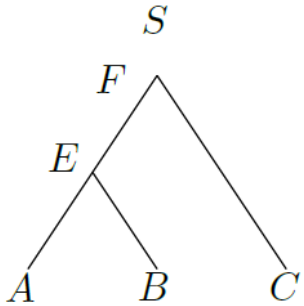
3 models

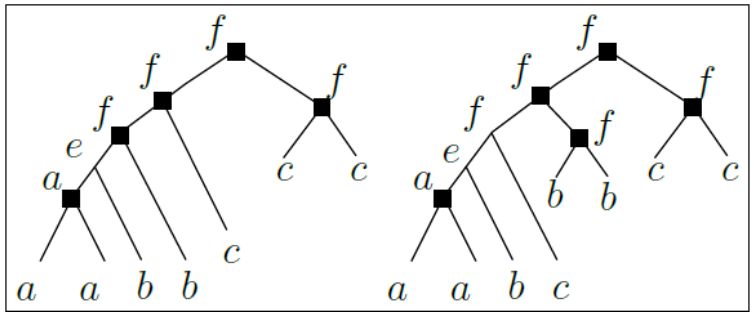
Episode Clustering



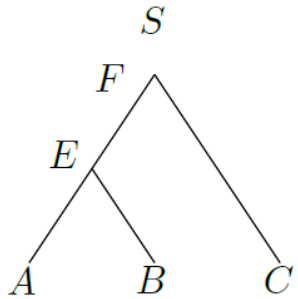


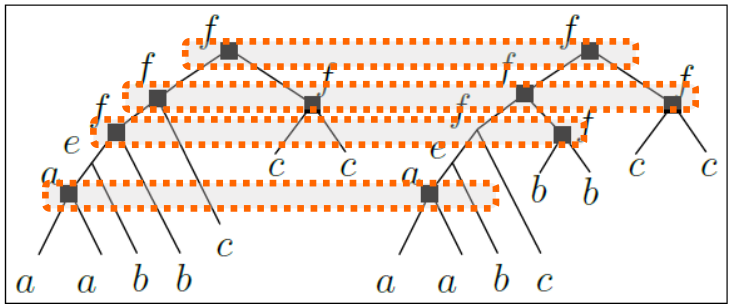
Gene Dup Clustering





Minimum Episode Clustering





Some more on Minimum Episode inference

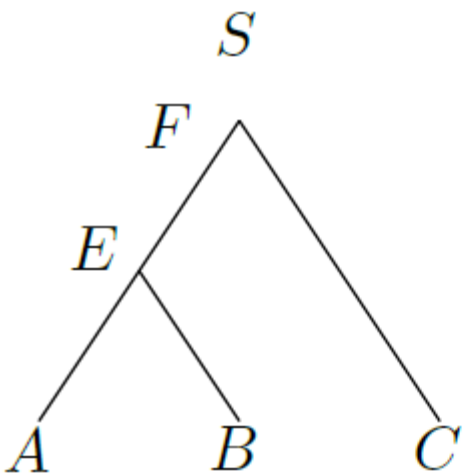
- Minimum Episodes inference problem
  - **Input** : species tree S, gene trees $G_1, \ldots, G_n$
  - **Find** : a valid gene-species mapping m that minimizes the number of ME duplications.

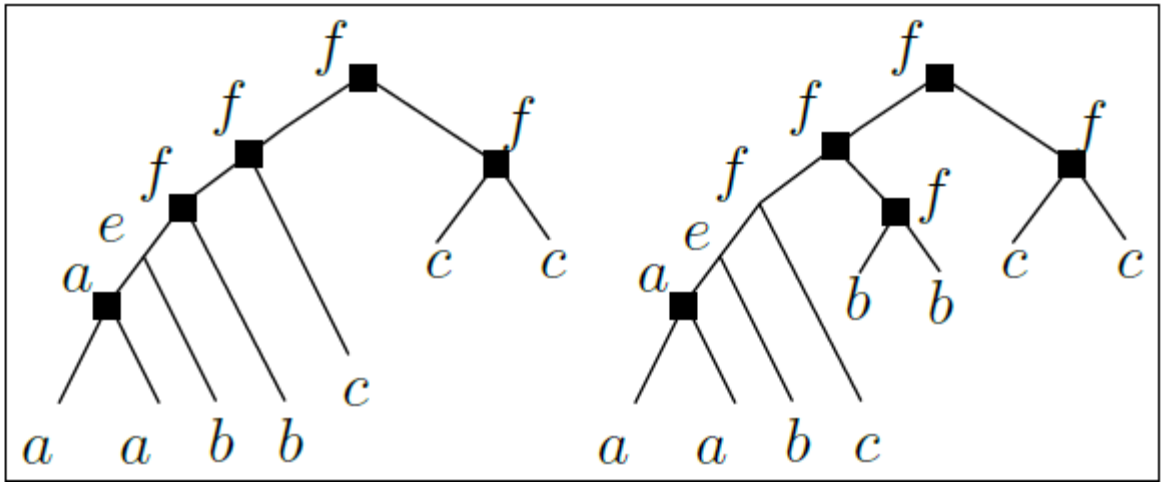
- Minimum Episodes inference problem
  - **Input** : species tree S, gene trees $G_1, \ldots, G_n$
  - **Find** : a valid gene-species mapping m that minimizes the number of ME duplications.

- Naive algorithm:
  - For each valid mapping m
    - Compute the number of ME duplications under m
  - Return the best mapping found

Reconciling with segmental Dups

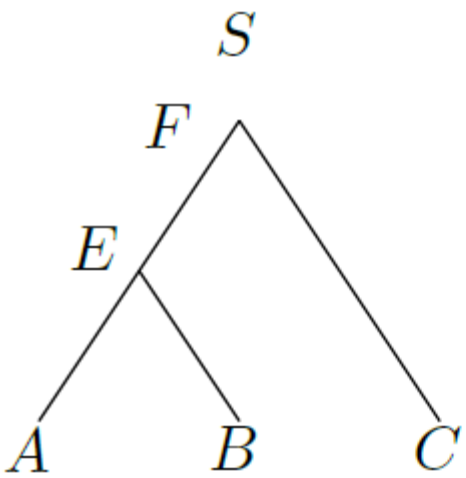
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?

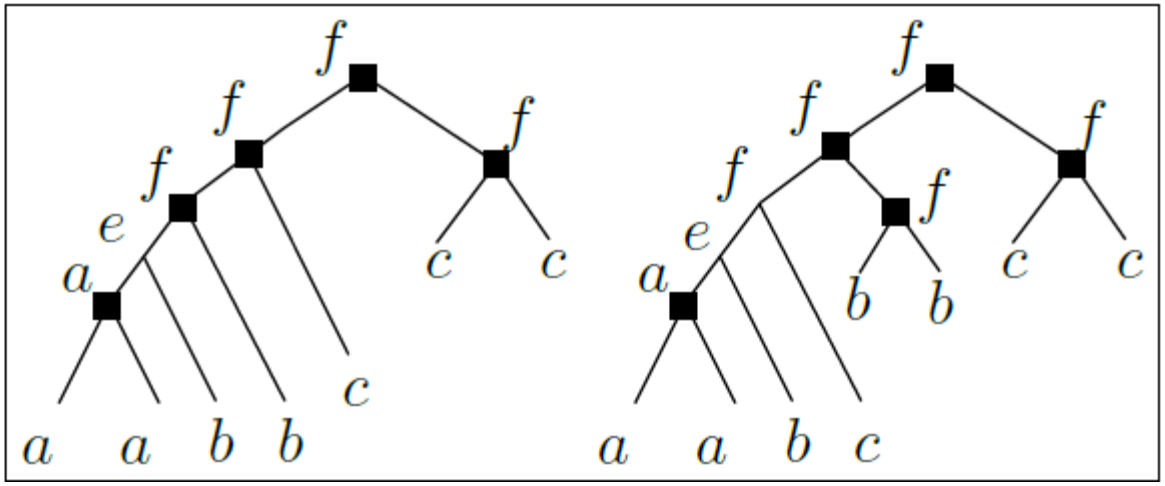




Reconciling with segmental Dups

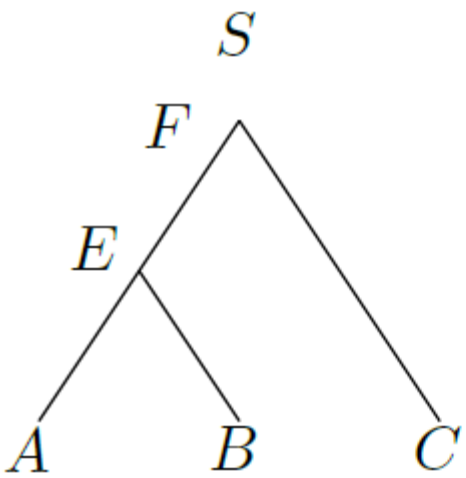
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

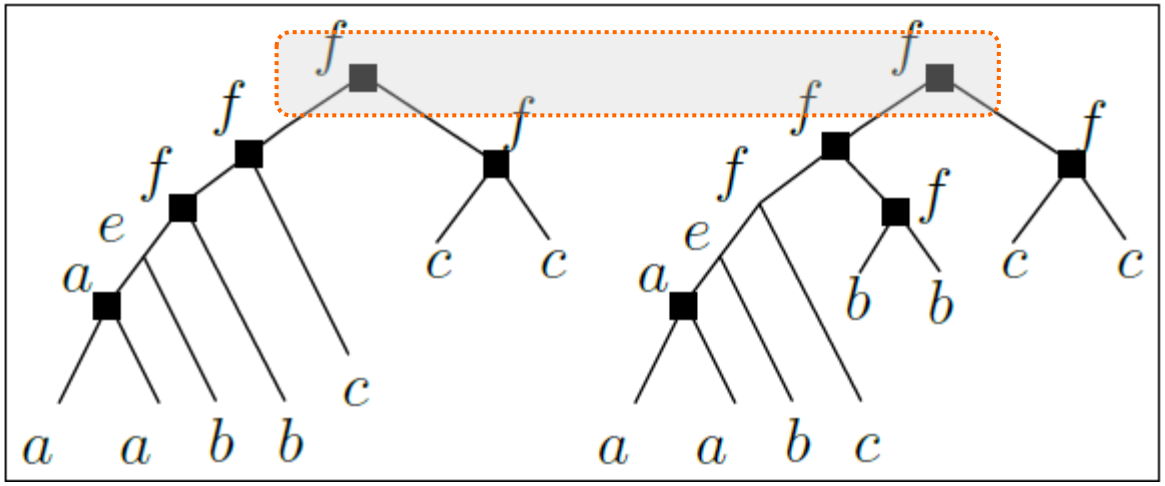




Reconciling with segmental Dups

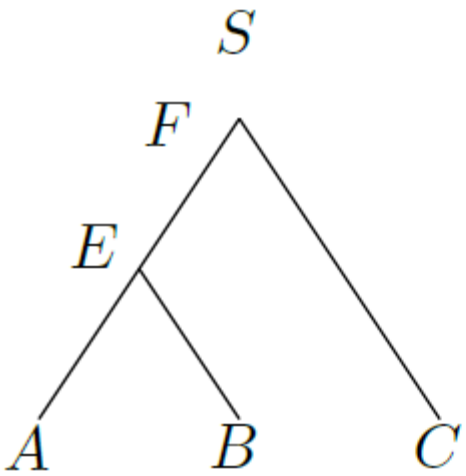
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

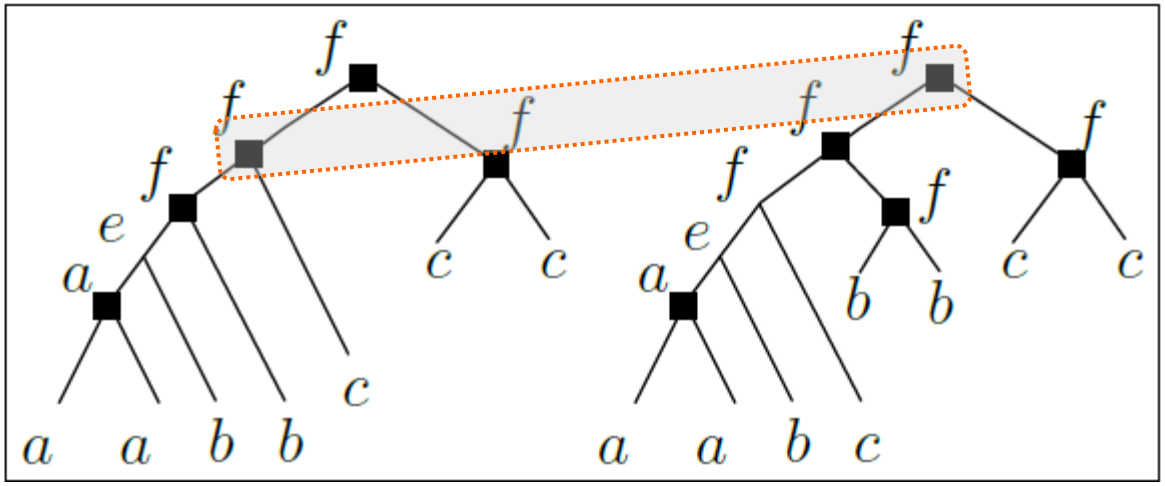




Reconciling with segmental Dups

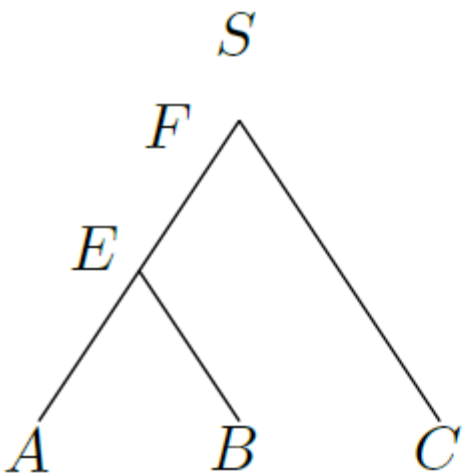
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

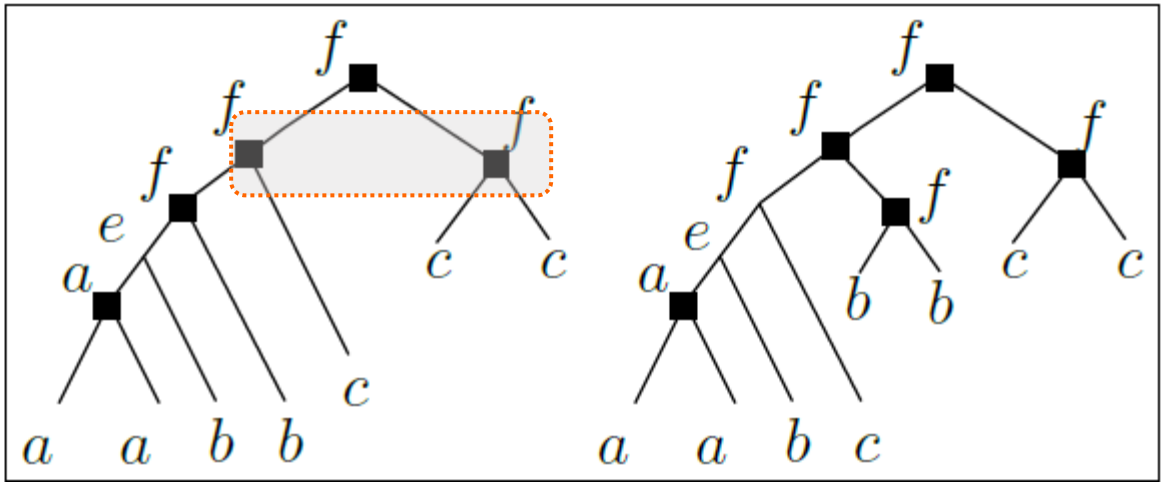




Reconciling with segmental Dups

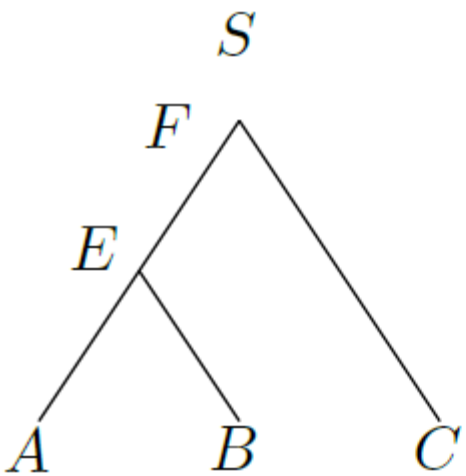
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

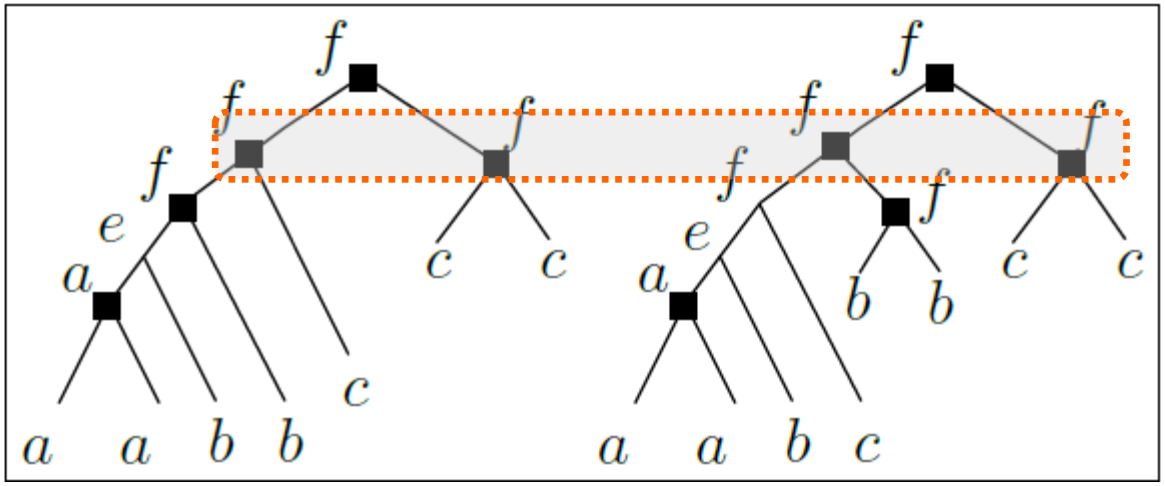




Reconciling with segmental Dups

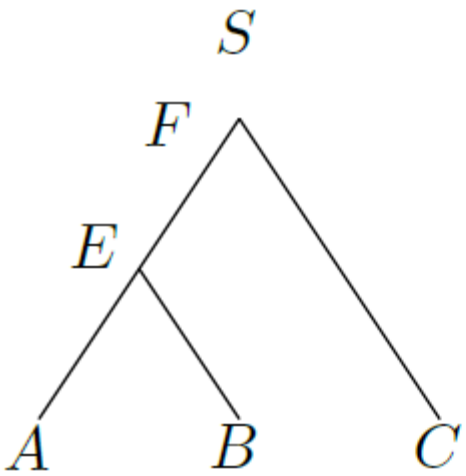
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

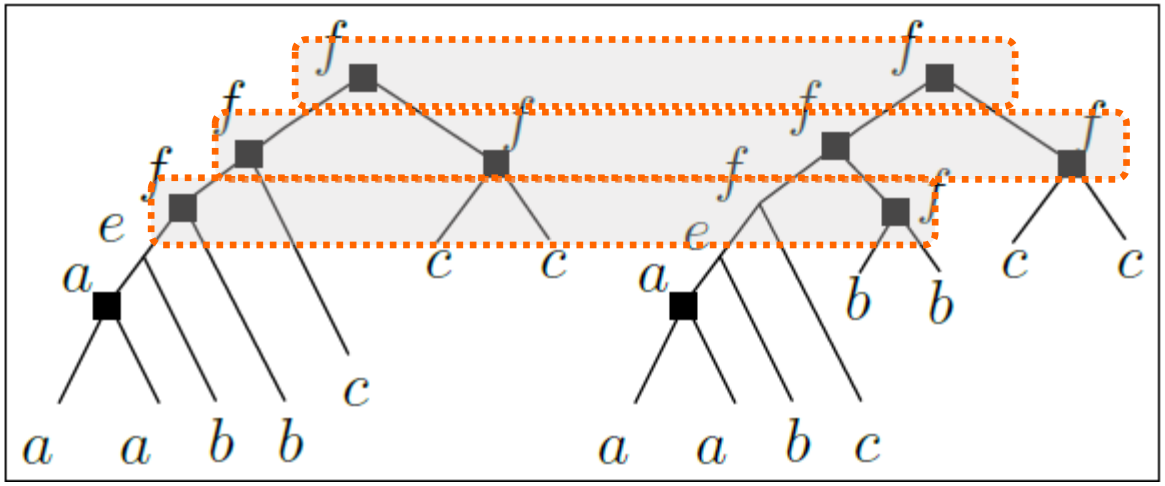




Reconciling with segmental Dups

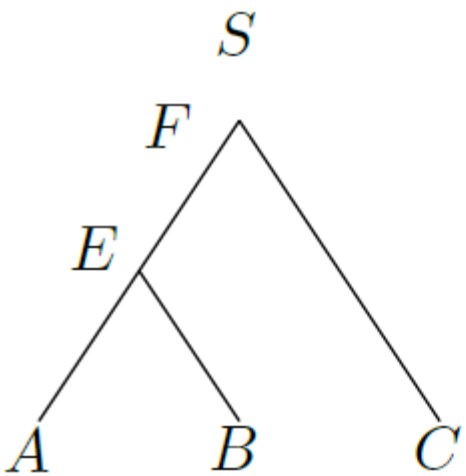
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

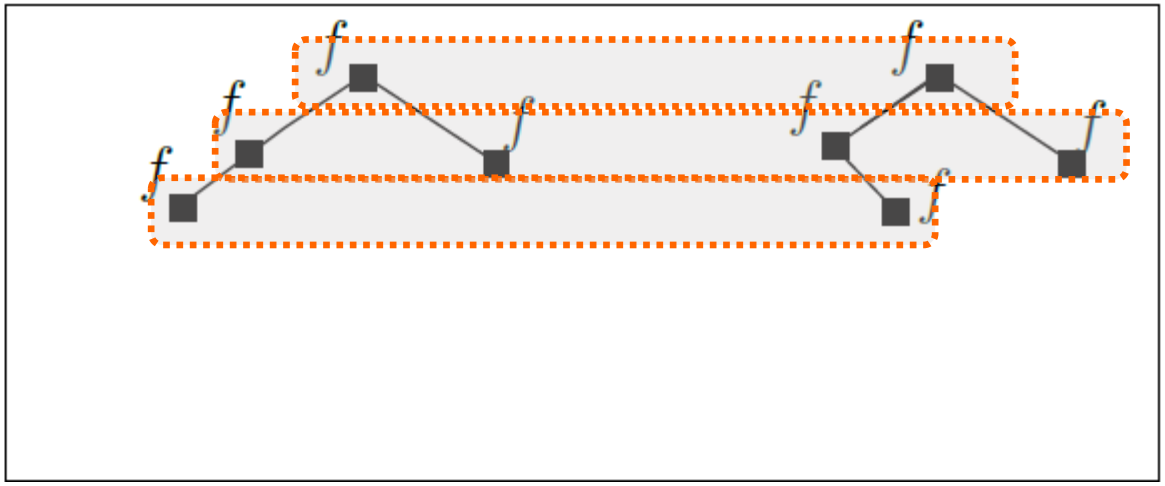




Reconciling with segmental Dups

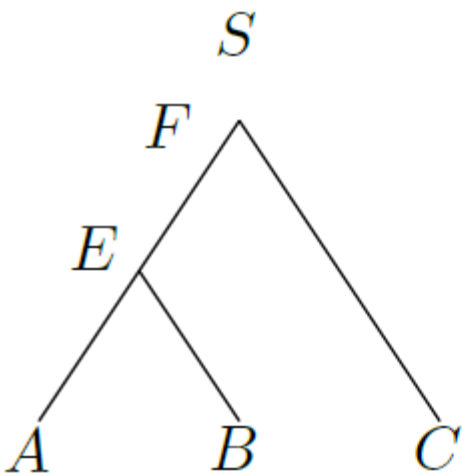
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.

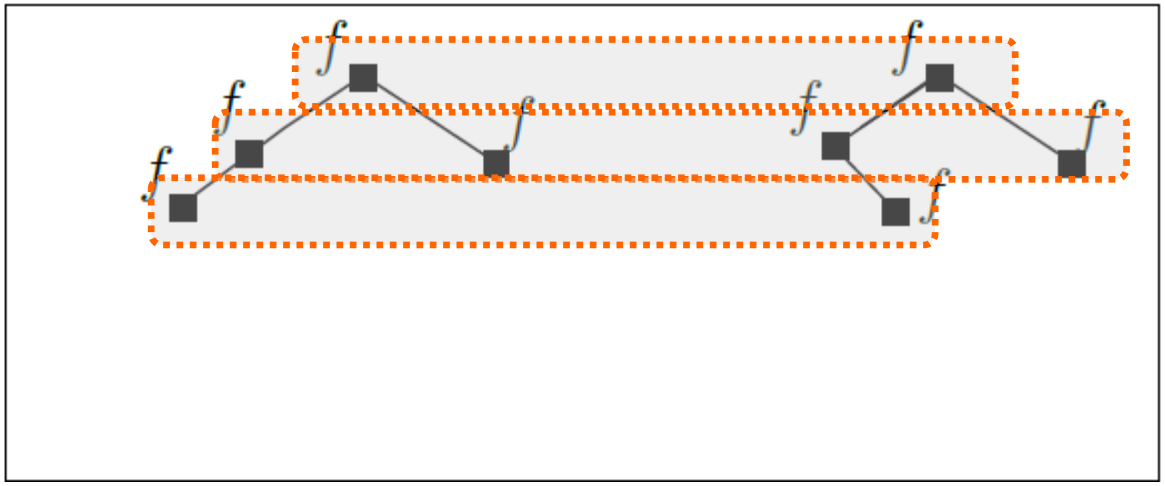




Reconciling with segmental Dups

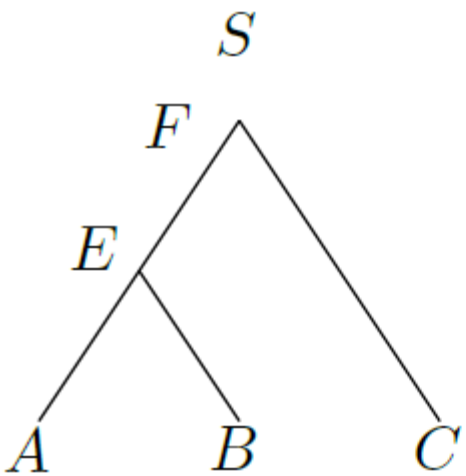
- **Question**: given a fixed mapping m, how do we minimize the number of ME Dups?
  - Dup events can affect genes in the same species, but **cannot contain a gene and one of its descendants**.
  - **# segmental Dups in *f* = height of *f* forest**

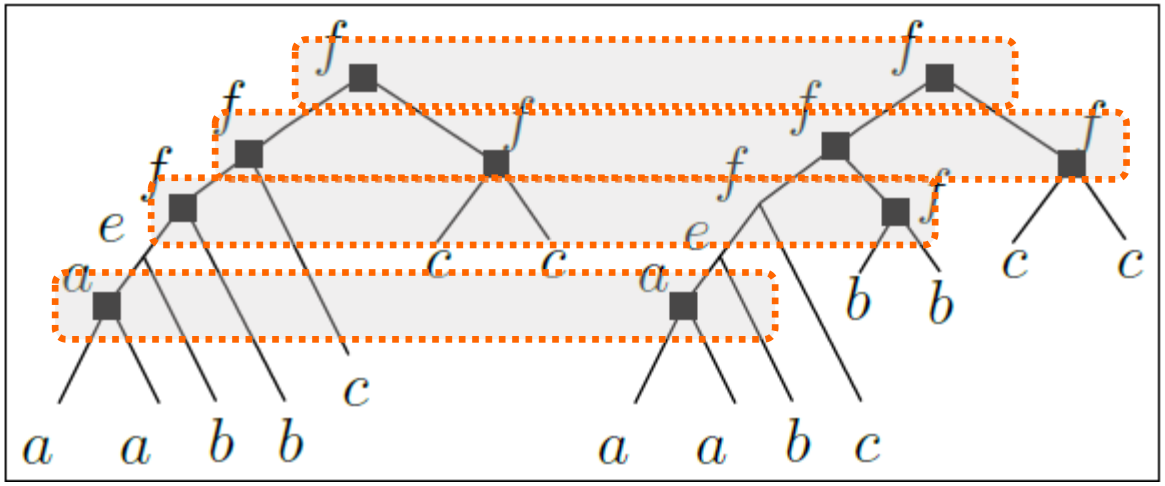




Reconciling with segmental Dups

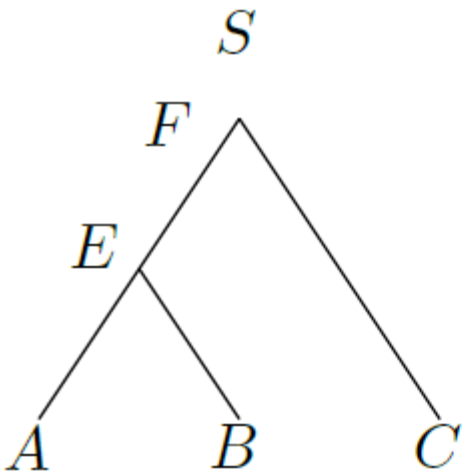
- **# segmental Dups in *f* = height of *f* forest = 3**
- **# segmental Dups in *a* = height of *a* forest = 1**
- **Total dup cost = 4**

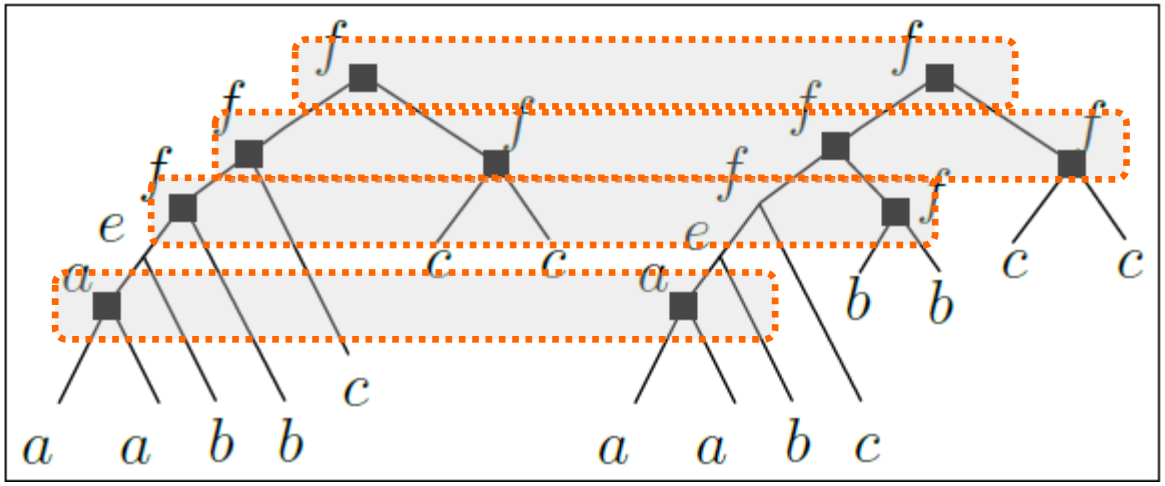




Reconciling with segmental Dups

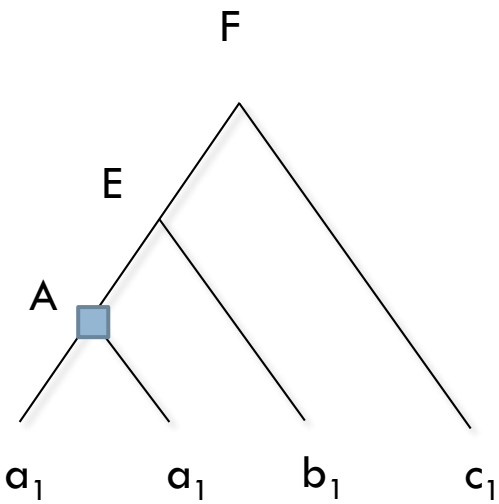
- **Input** : species tree S, gene trees $G_1, \ldots, G_n$
- **Find** : a valid gene-species mapping m that minimizes the sum of dup heights $\sum_{v \in V(S)} dupheight(v)$.

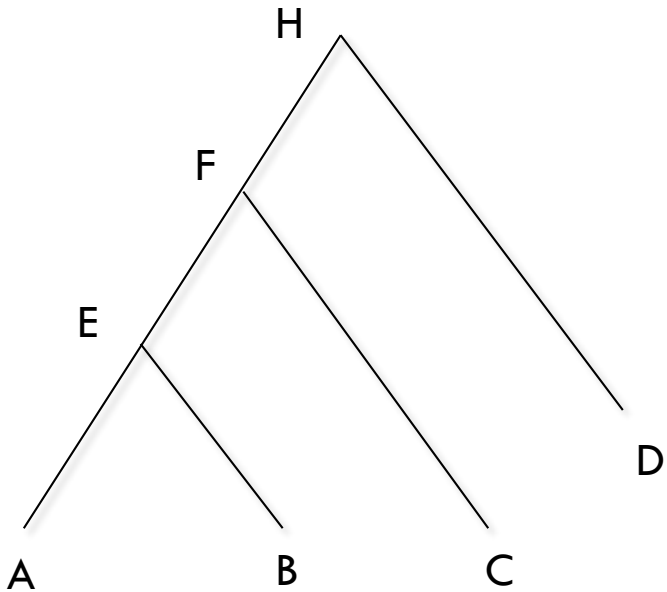




Reconciling with segmental Dups

- Main difficulty : remapping a Dup can create a chain of Dups above it.

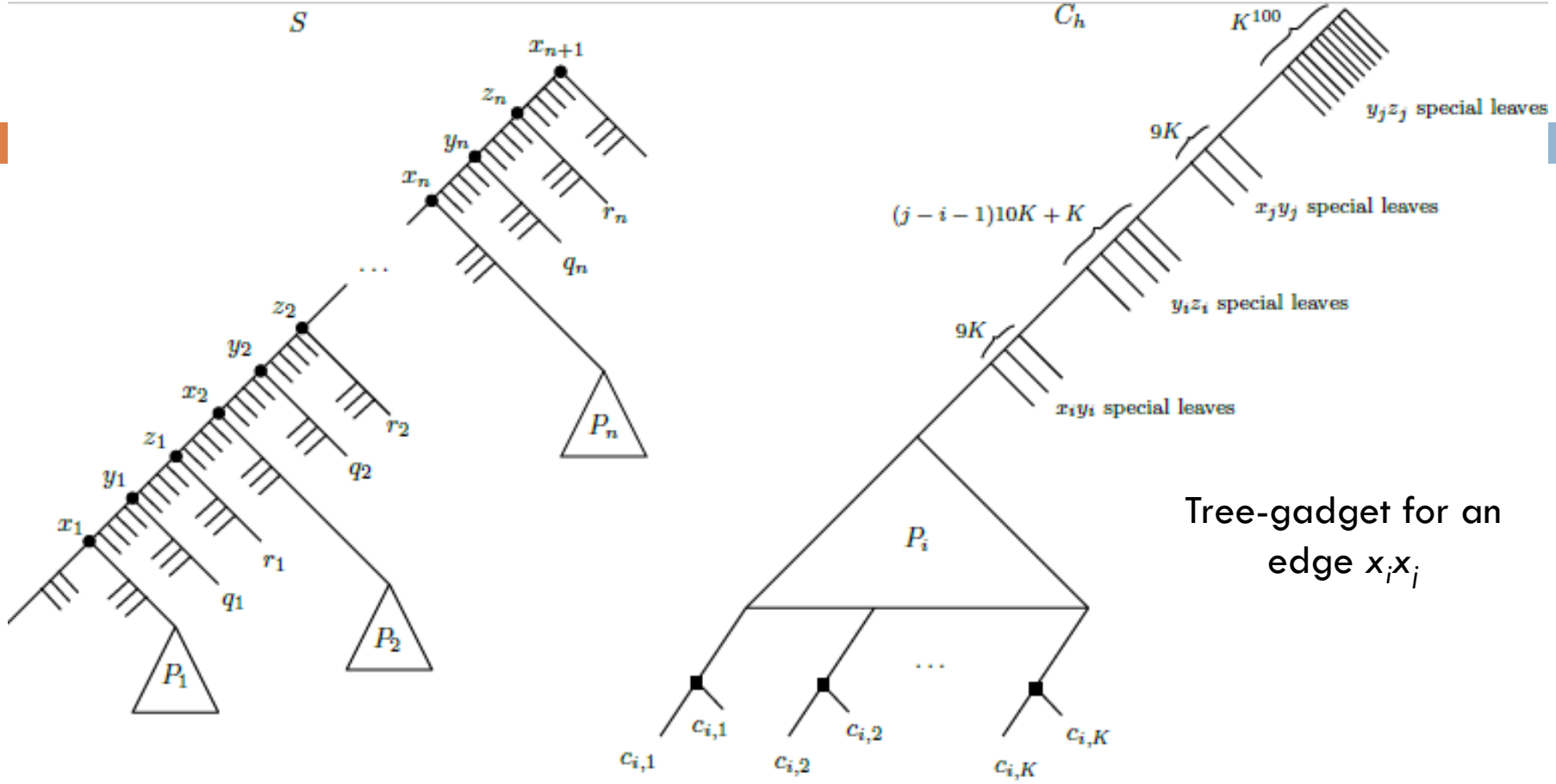




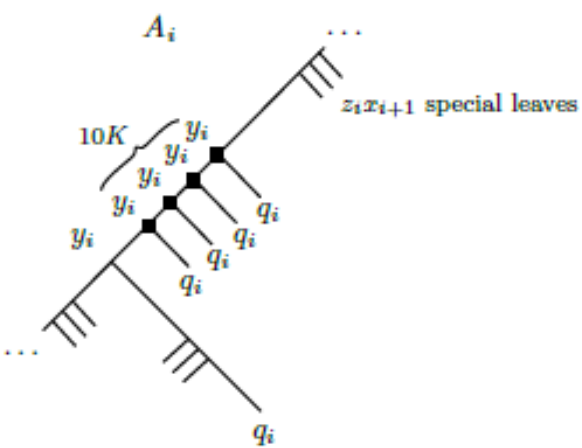
NP-hardness of ME clustering

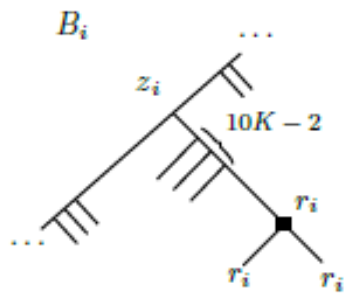
- Complexity was left opened in Paszek & Gorecki in 2017.

- **Theorem:** Finding an optimal reconciliation with the minimum number of ME Dups is NP-hard.
  - *[Dondi, L & Scornavacca, AMB 2019]*
  - Reduction from Vertex Cover



Tree-gadget for an edge $x_i x_j$



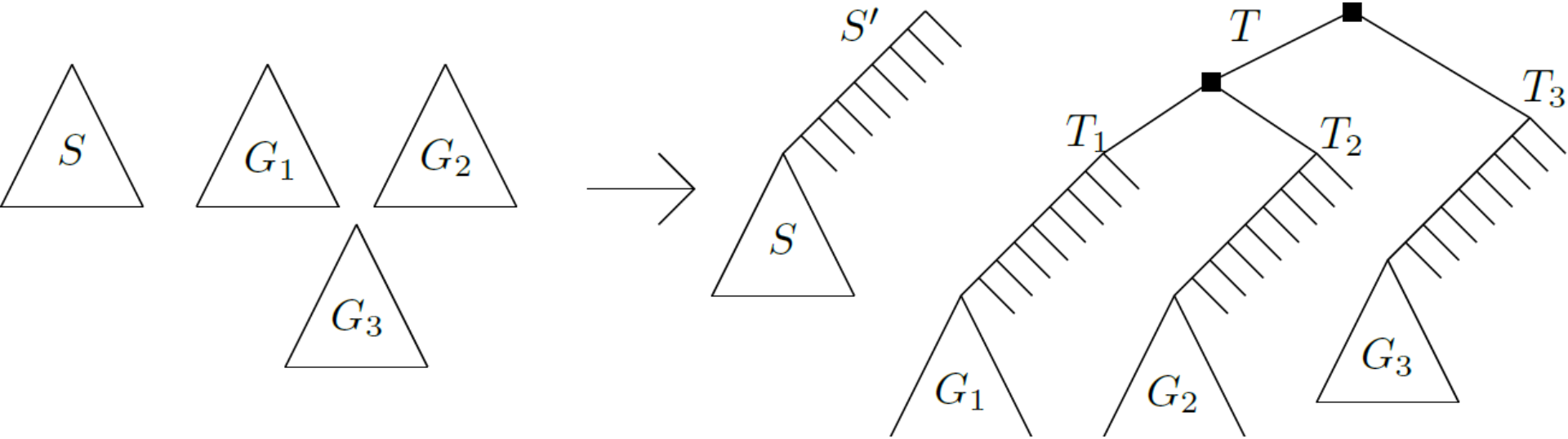


NP-hardness of ME clustering

- **Theorem**: finding an optimal reconciliation with minimum ME Dups is NP-hard, **even if only one gene tree is given in the input**.

NP-hardness of ME clustering

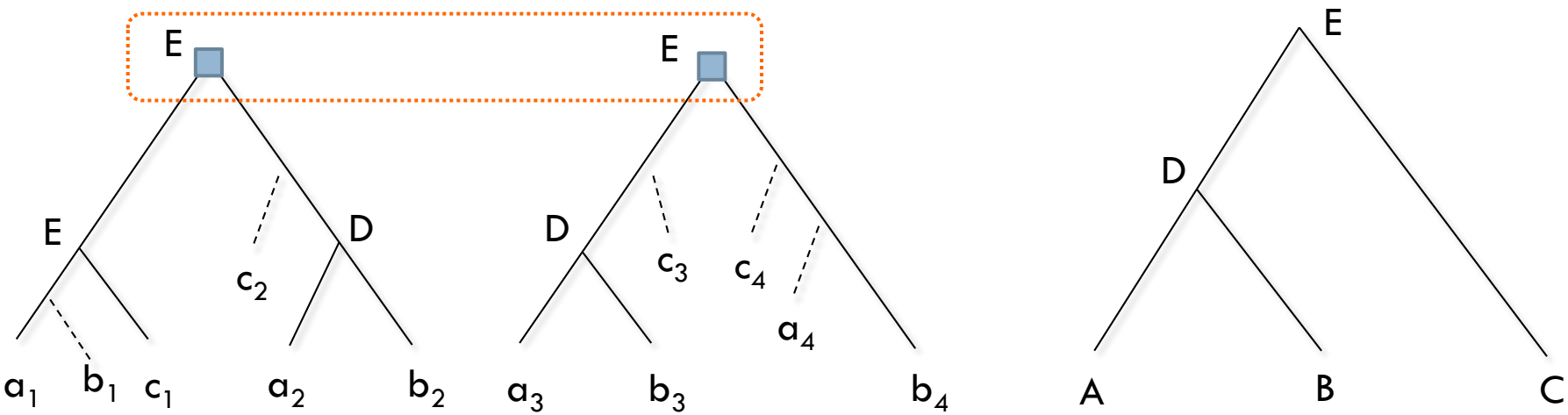
- **Theorem**: finding an optimal reconciliation with minimum ME Dups is NP-hard, **even if only one gene tree is given in the input.**
  - Reduction from reconciliation with many gene trees: just join all the gene trees under many speciations.



Incorporating gene losses

Incorporating gene losses

- **Input** : species tree S, gene trees $G_1, \ldots, G_n$, dup cost δ, loss cost λ
- **Find** : a valid gene-species mapping m that minimizes δ * (sum of Dup heights) + λ * (number of losses)



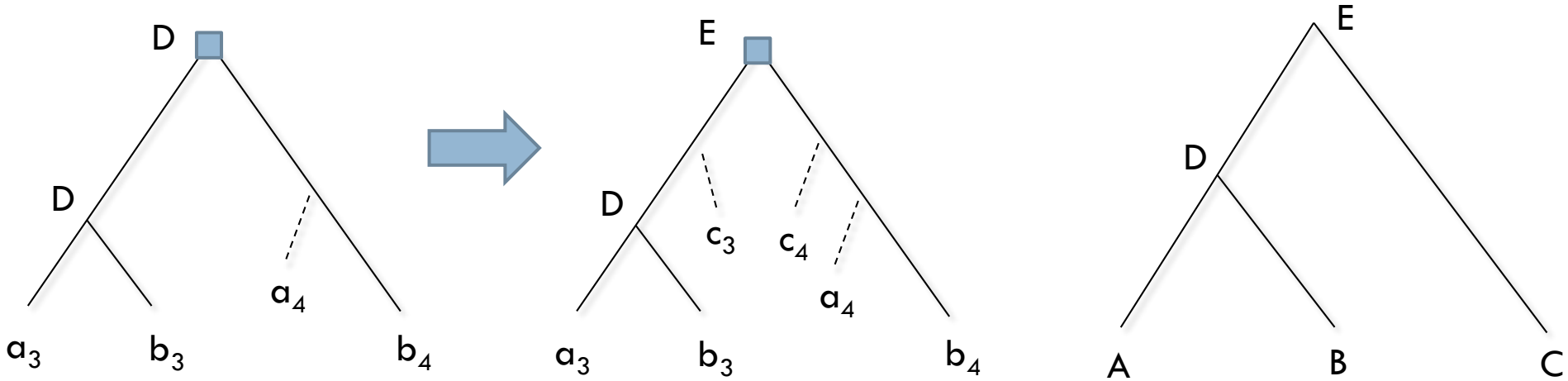
1 DUP, 5 LOSSES (before, we had 2 DUPS, 3 LOSSES)

The case of $\lambda \geq \delta$

- $\lambda \geq \delta$ => losses are worse than Dups.

The case of $\lambda \geq \delta$

- $\lambda \geq \delta$ => losses are worse than Dups.
- **Theorem**: when $\lambda \geq \delta$, the usual LCA mapping yields an optimal reconciliation. It is also the unique optimal reconciliation if $\lambda > \delta$.

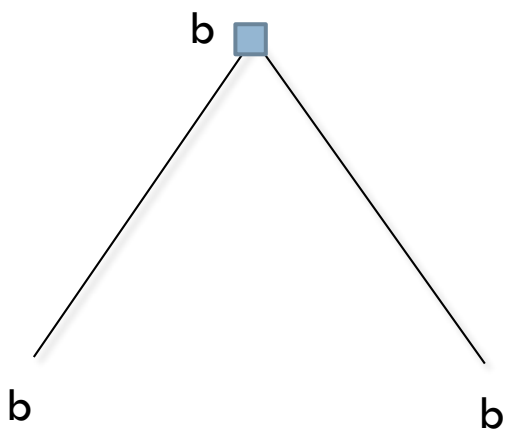


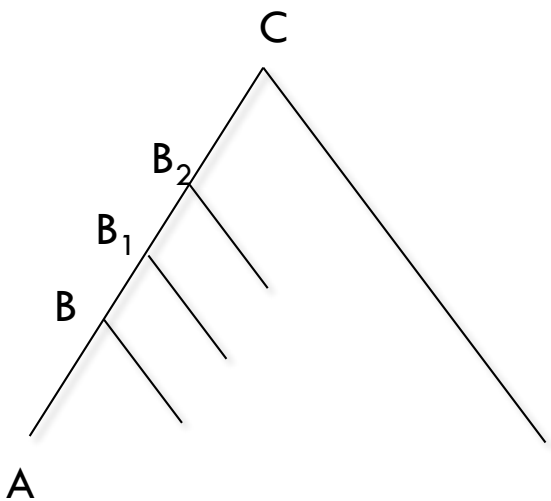
An FPT algorithm for $\lambda < \delta$

- An $O((\delta/\lambda)^{d+1} n)$ time algorithm.
  - *d* is the sum of Dup heights in an optimal solution
  - e.g. when $\delta = 3$, $\lambda = 2$, we get a $O(1.5^{d+1} n)$ algorithm.

An FPT algorithm for $\lambda < \delta$

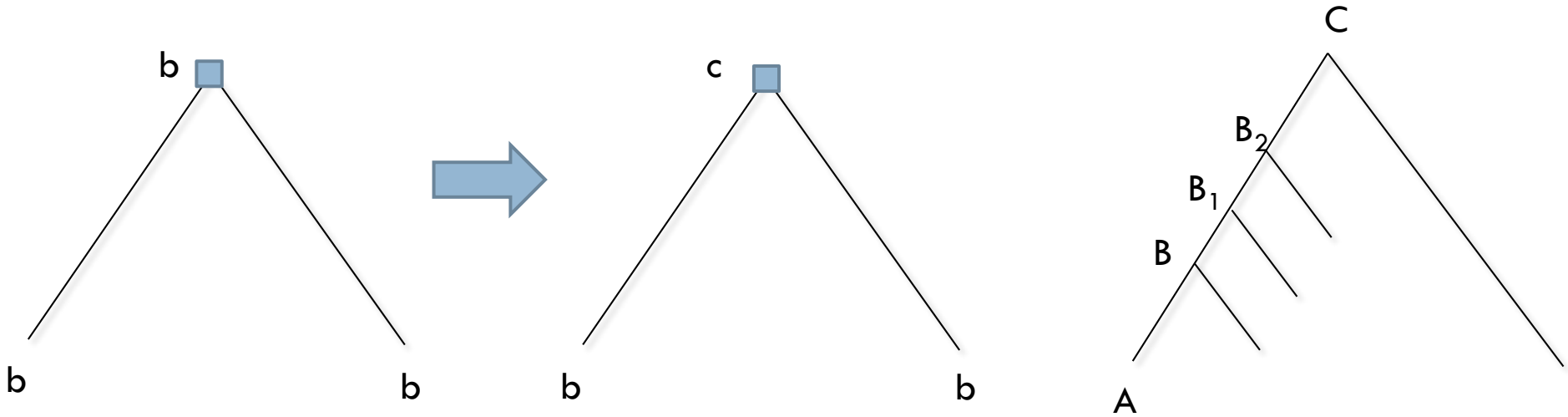
- When we remap a Dup node up by *k* species, we create at least *k* new losses.





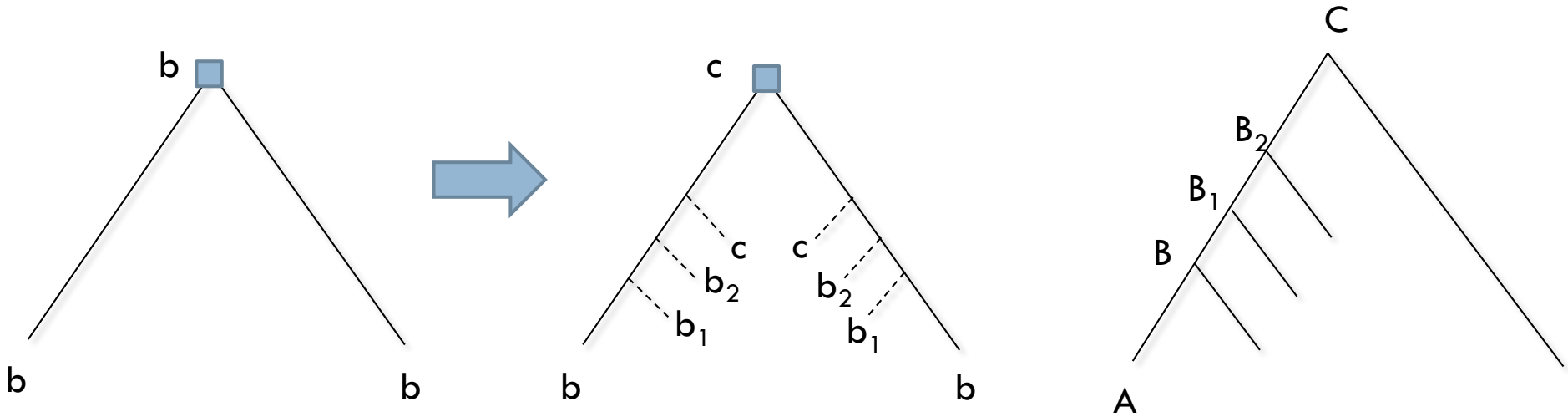
An FPT algorithm for $\lambda < \delta$

- When we remap a Dup node up by *k* species, we create at least *k* new losses.



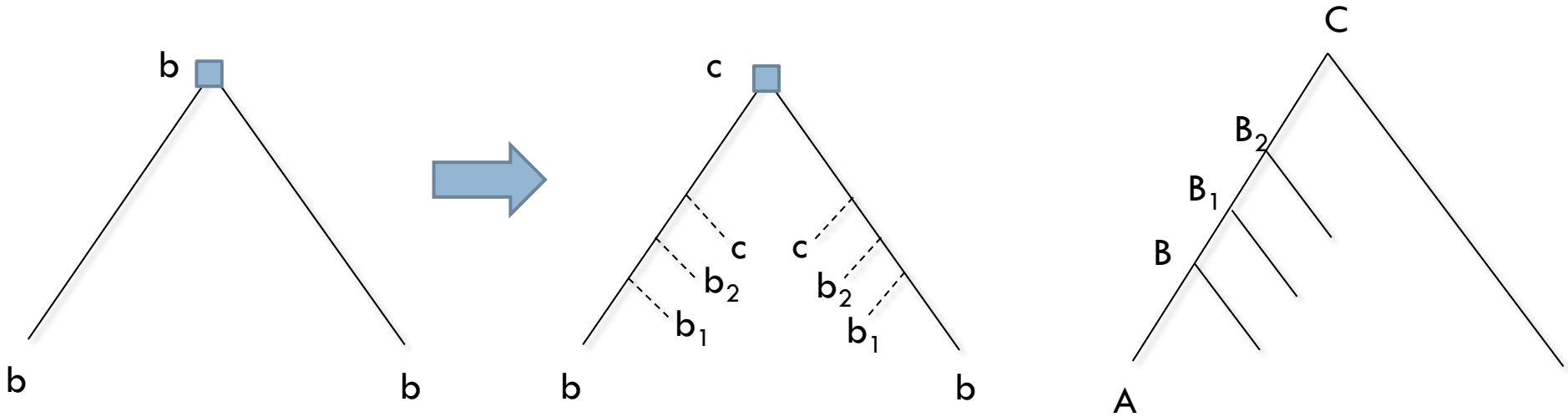
An FPT algorithm for $\lambda < \delta$

- When we remap a Dup node up by *k* species, we create at least *k* new losses.



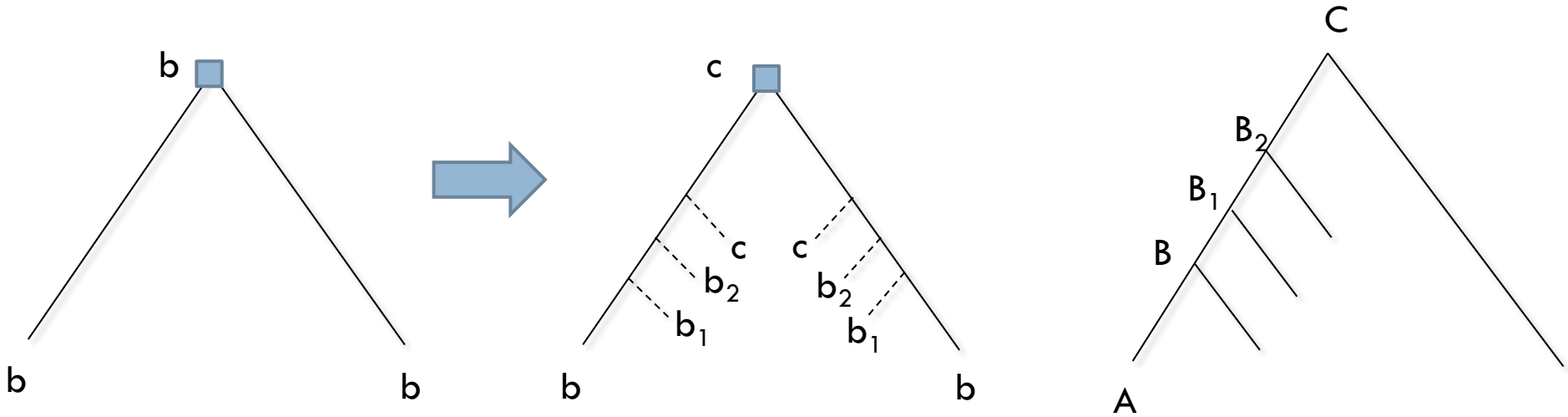
An FPT algorithm for $\lambda < \delta$

- When we remap a Dup node up by *k* species, we create at least *k* new losses.
- If we remap a Dup node up by more than δ/λ species, we save 1 Dup but create $> \delta/\lambda$ losses.



An FPT algorithm for $\lambda < \delta$

- When we remap a Dup node up by *k* species, we create at least *k* new losses.
- If $k > \delta/\lambda$ losses, *never worth it*.



An FPT algorithm for $\lambda < \delta$

- Branching algorithm:
  - Take a Dup node *x* mapped to species *s* under the LCA mapping.
  - Branch into the δ/λ possible ways of remapping *x* to an ancestor *s'* of *s*.
    - *If x is well-chosen, each time we branch, Dup heights increase by 1.*

An FPT algorithm for $\lambda < \delta$

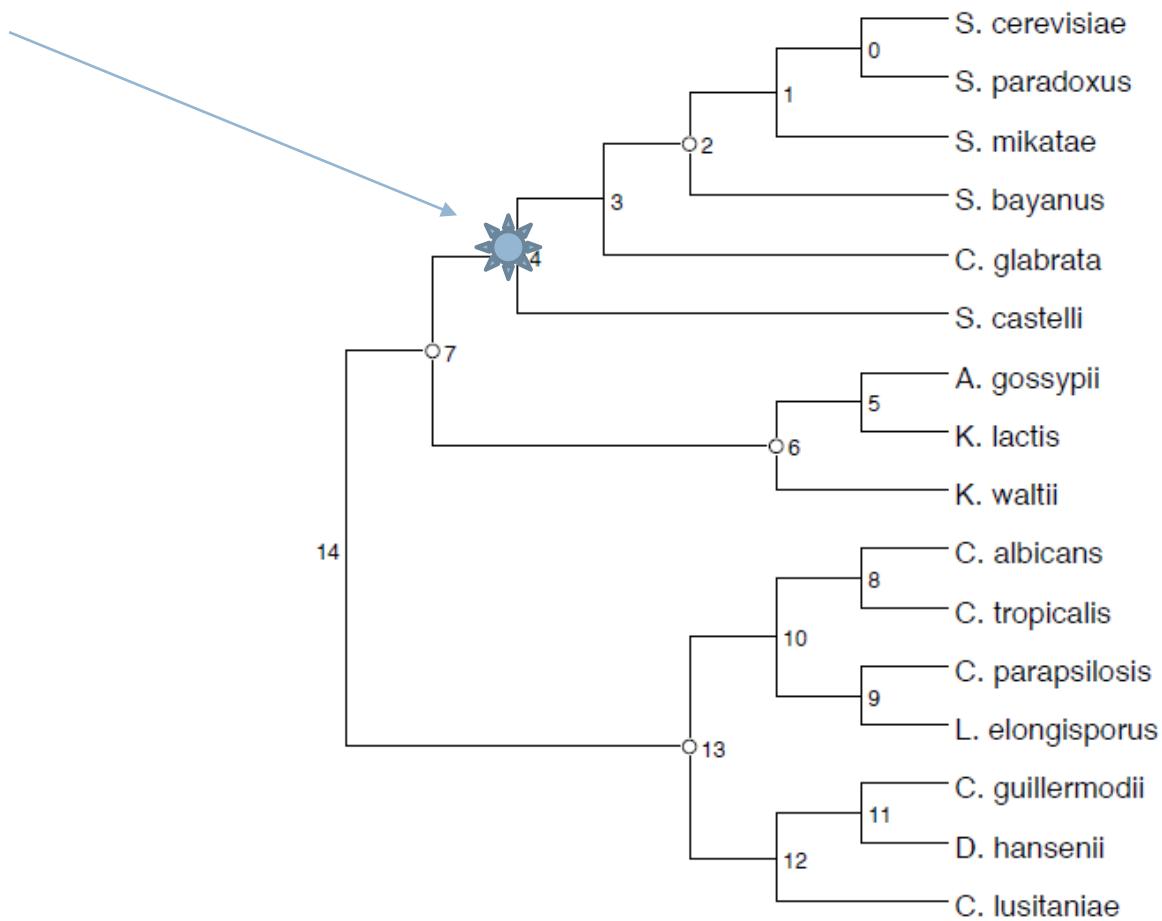
- Branching algorithm:
  - Take a Dup node *x* mapped to species *s* under the LCA mapping.
  - Branch into the δ/λ possible ways of remapping *x* to an ancestor *s'* of *s*.
    - *If x is well-chosen, each time we branch, Dup heights increase by 1.*
  - Search tree of degree δ/λ and height at most *d*.
    - $O((\delta/\lambda)^{d+1} n)$ complexity

Experiments

- We implemented the FPT algorithm.
  - https://github.com/manuellafond/Multrec
- We applied it on 2 datasets:
  - Yeast species from [Butler & al., Nature, 2009]
    - 16 species, 2379 gene trees
  - Eukaryotes from [Guigo & al., Mol Phylo Evo, 1996]
    - 16 species, 53 gene trees

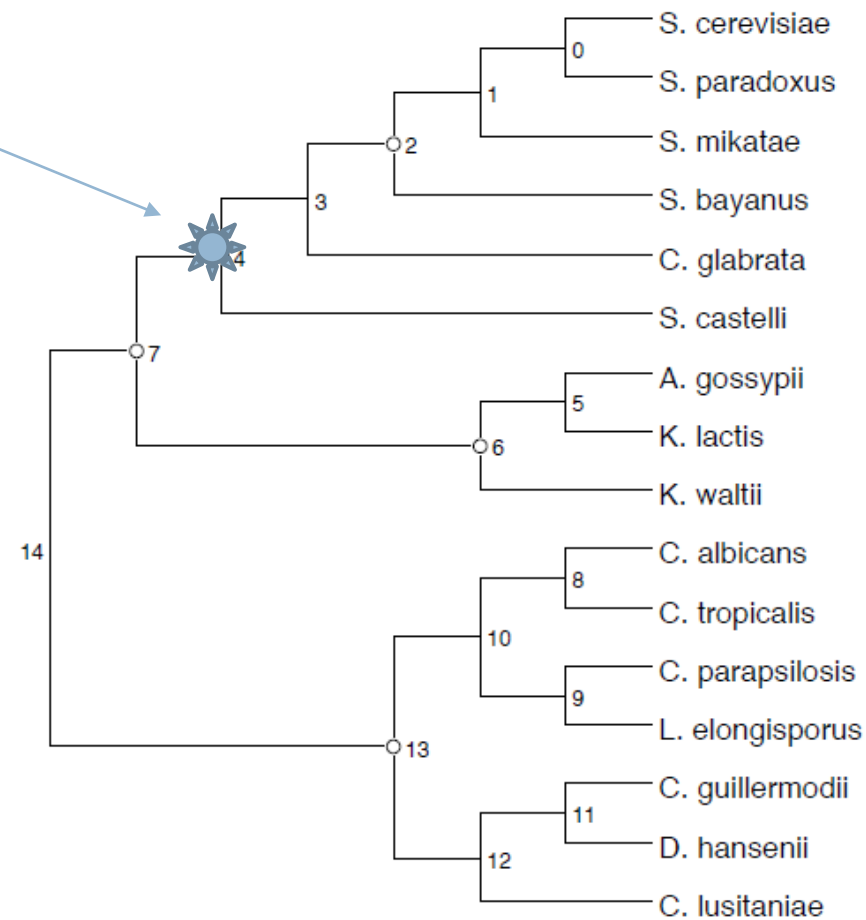
Experiments

- In the 2379 yeast trees, we infer a segmental Dup with **216 genes** ($\delta = 3, \lambda = 2$).
  - Located here



Experiments

- In the 2379 yeast trees, we infer a segmental Dup with **216 genes** ($\delta = 3, \lambda = 2$).
  - Located here
  - Coincides with WGD found using synteny in *[Kellis, Birren & Lander, Nature, 2004]*



Nodes 7,6,13,2 had segmental Dup with 190, 157, 148 and 136 *genes.*

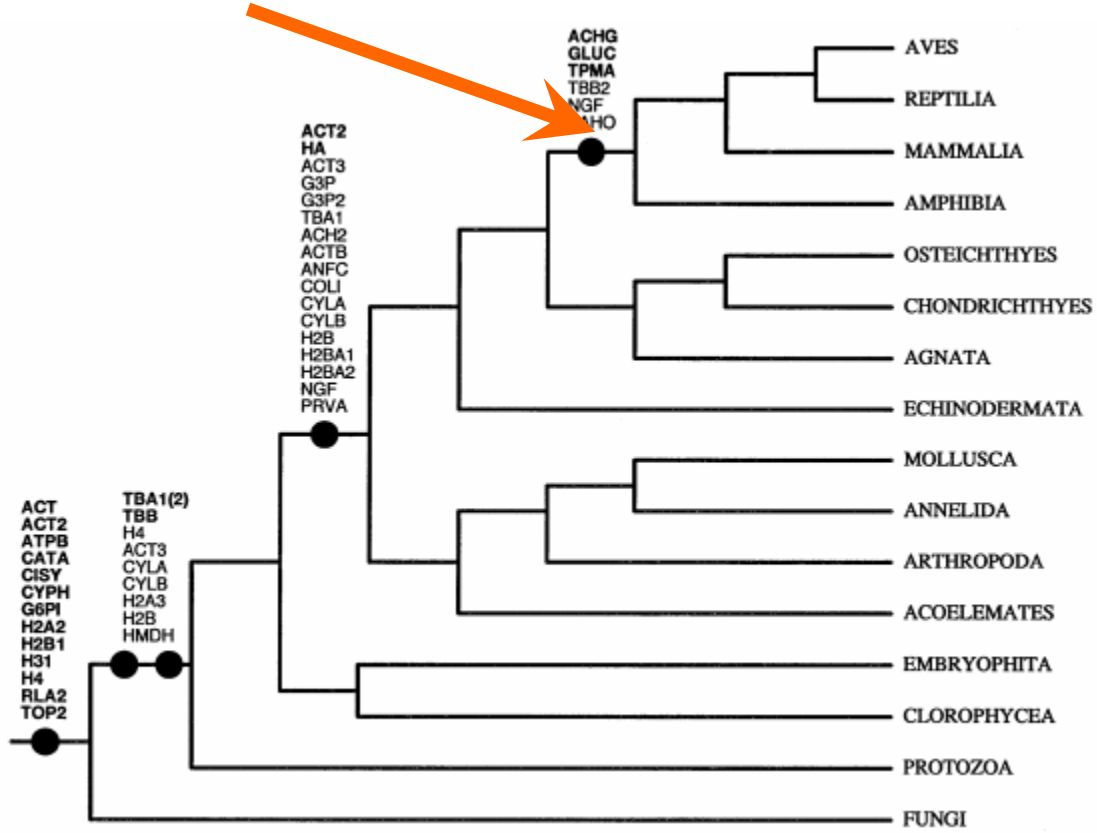
Experiments

- In the 53 Eukaryote gene trees.
  - *ExactMGD [Bansal & Eulenstein, Bioinf, 2008]* finds a solution with **5** segmental Dups
    - Does not allow speciations to become duplications.
  - We find a solution with **4 segmental Dups**
    - By setting $\delta > 61$, $\lambda = 1$
    - All segmental Dups found in *[Guigo & al., 1996]* are confirmed, **EXCEPT ONE**.

Experiments

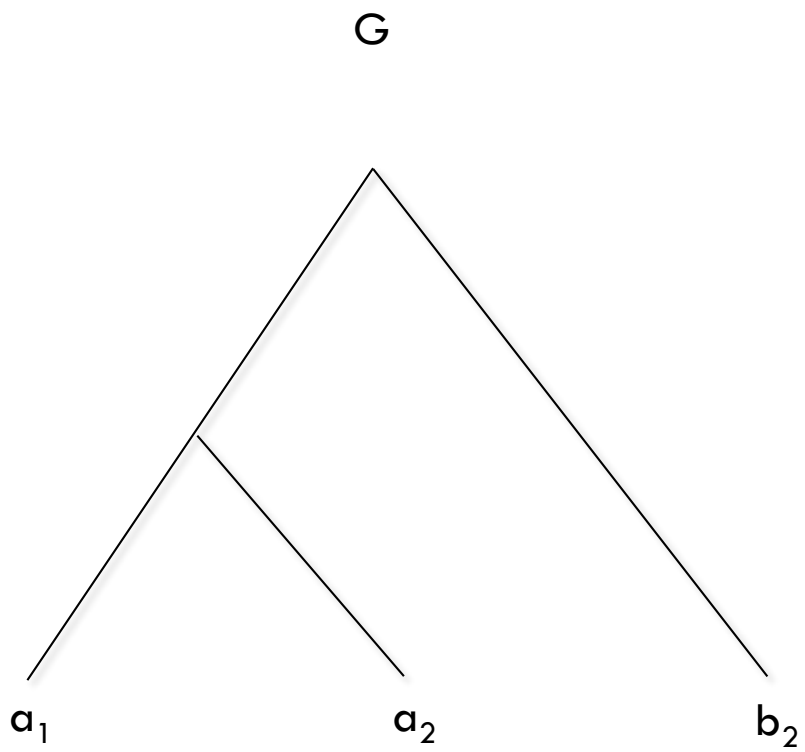
- In the 53 Eukaryote gene trees.

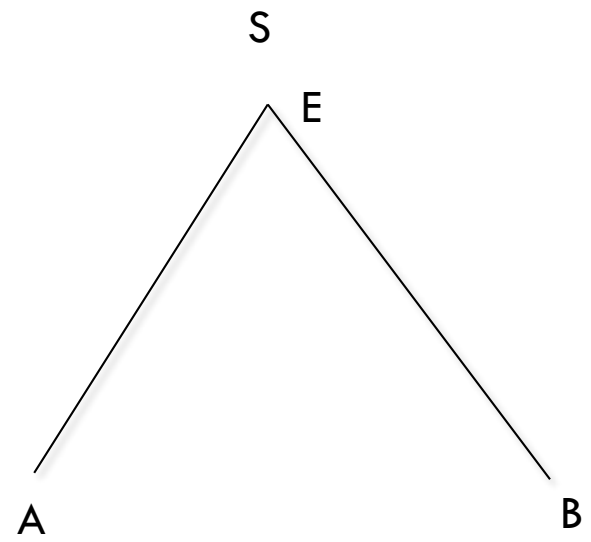
In our solutions, no Dup maps here (Tetrapoda)

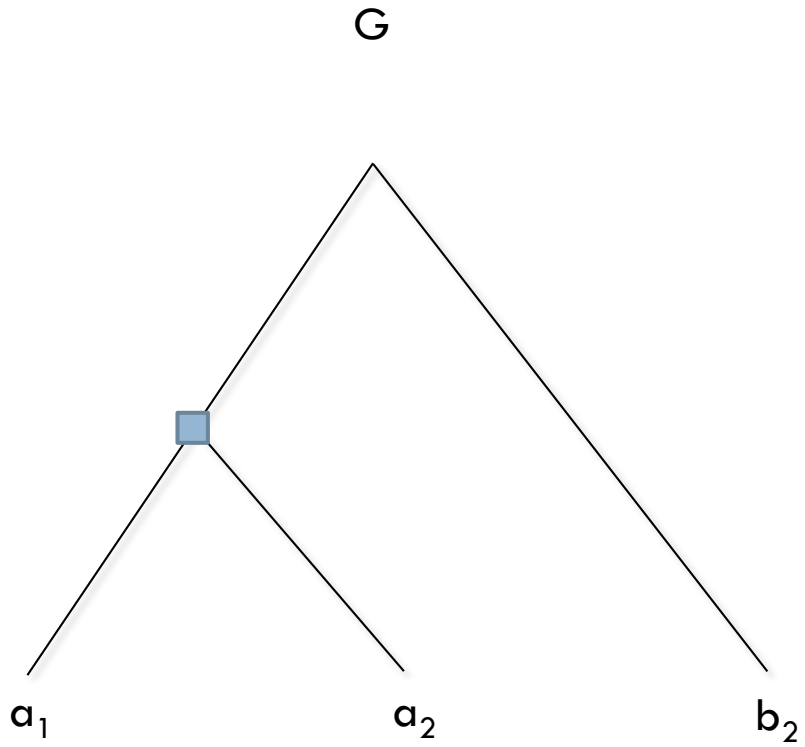


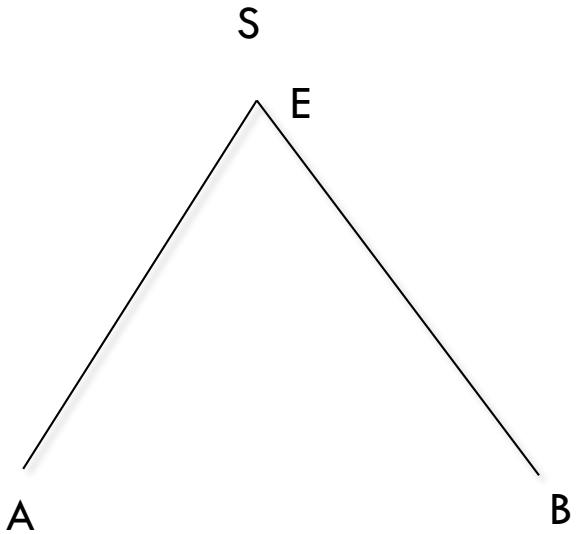
- Algorithmic challenges
  - Without losses, is the problem FPT in d? And with losses?
  - Constant factor approximation?
- Modeling challenges
  - Model the problem with segmental dups + segmental losses
  - Add segmental horizontal gene transfers
  - Add conserved adjacencies / syntenies into the optimization criteria

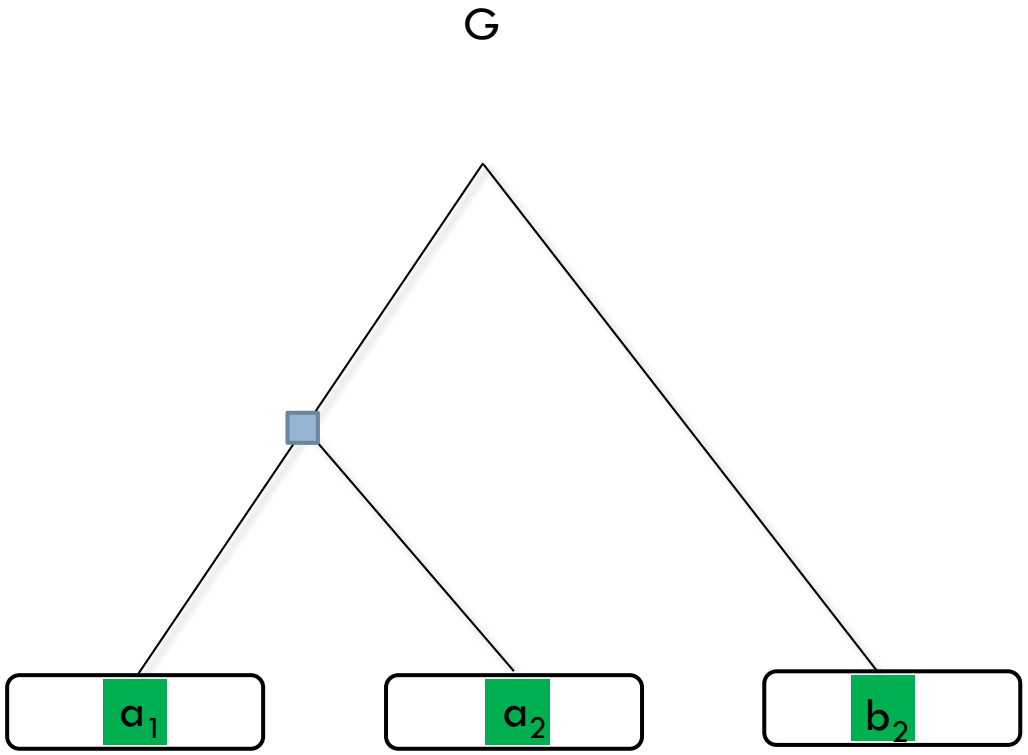
Incorporating syntenic blocks

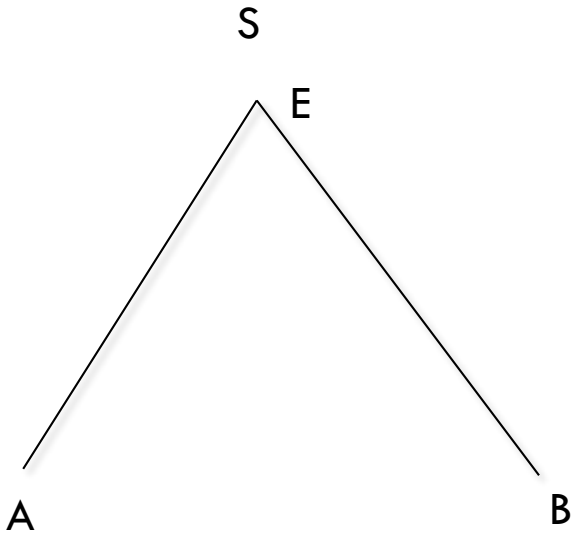

G
a_1
a_2
b_2

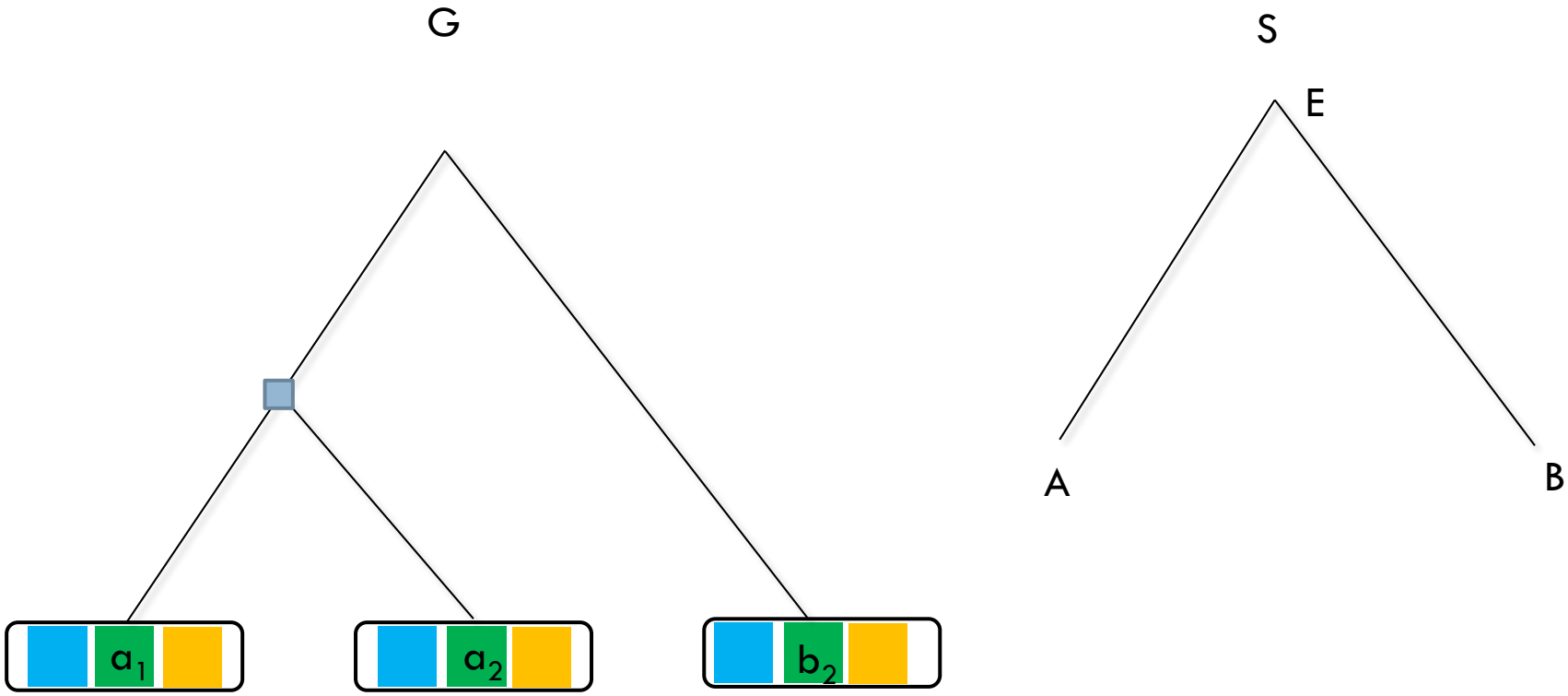

S
E
A
B

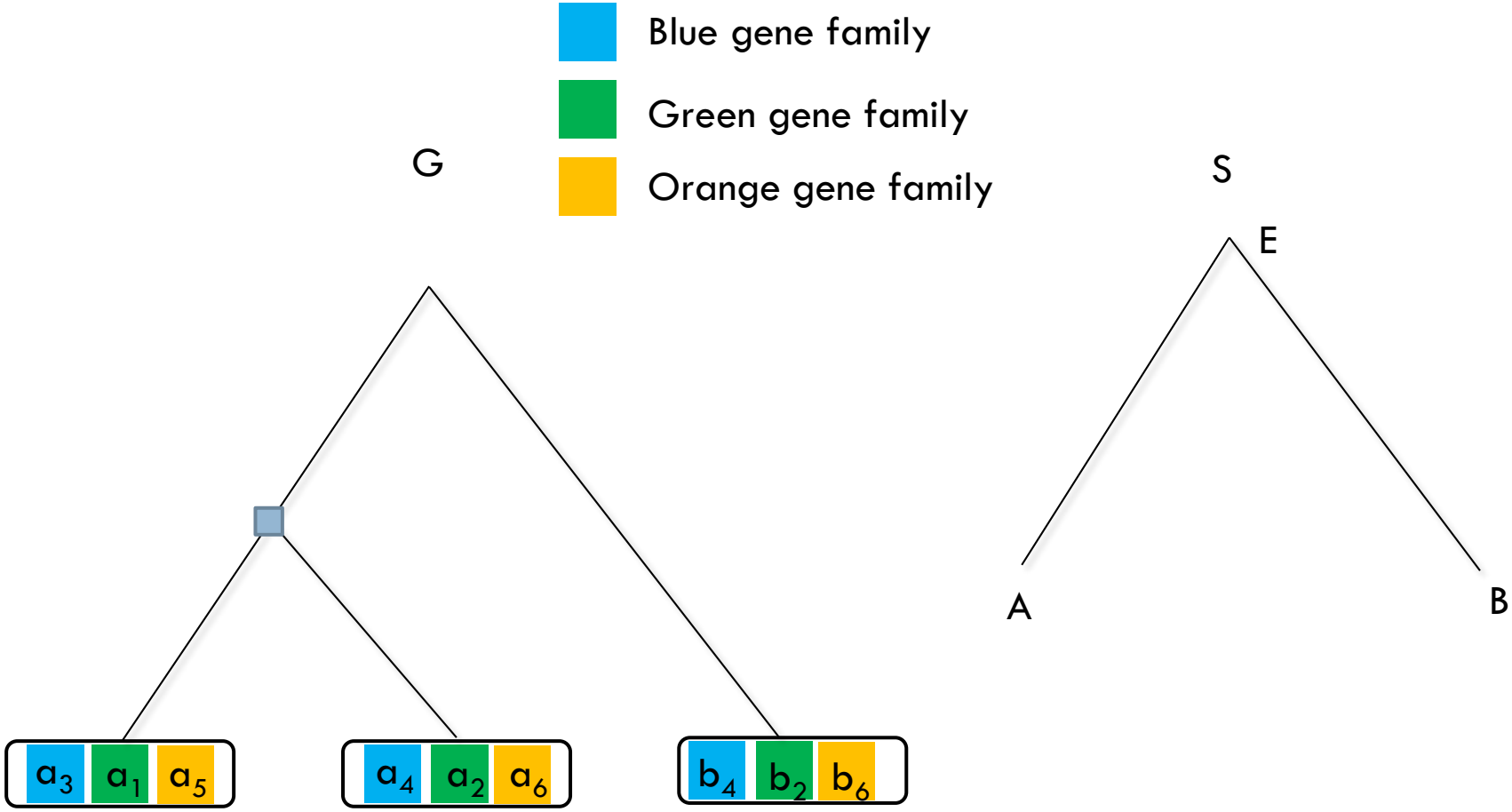

G
a_1
a_2
b_2

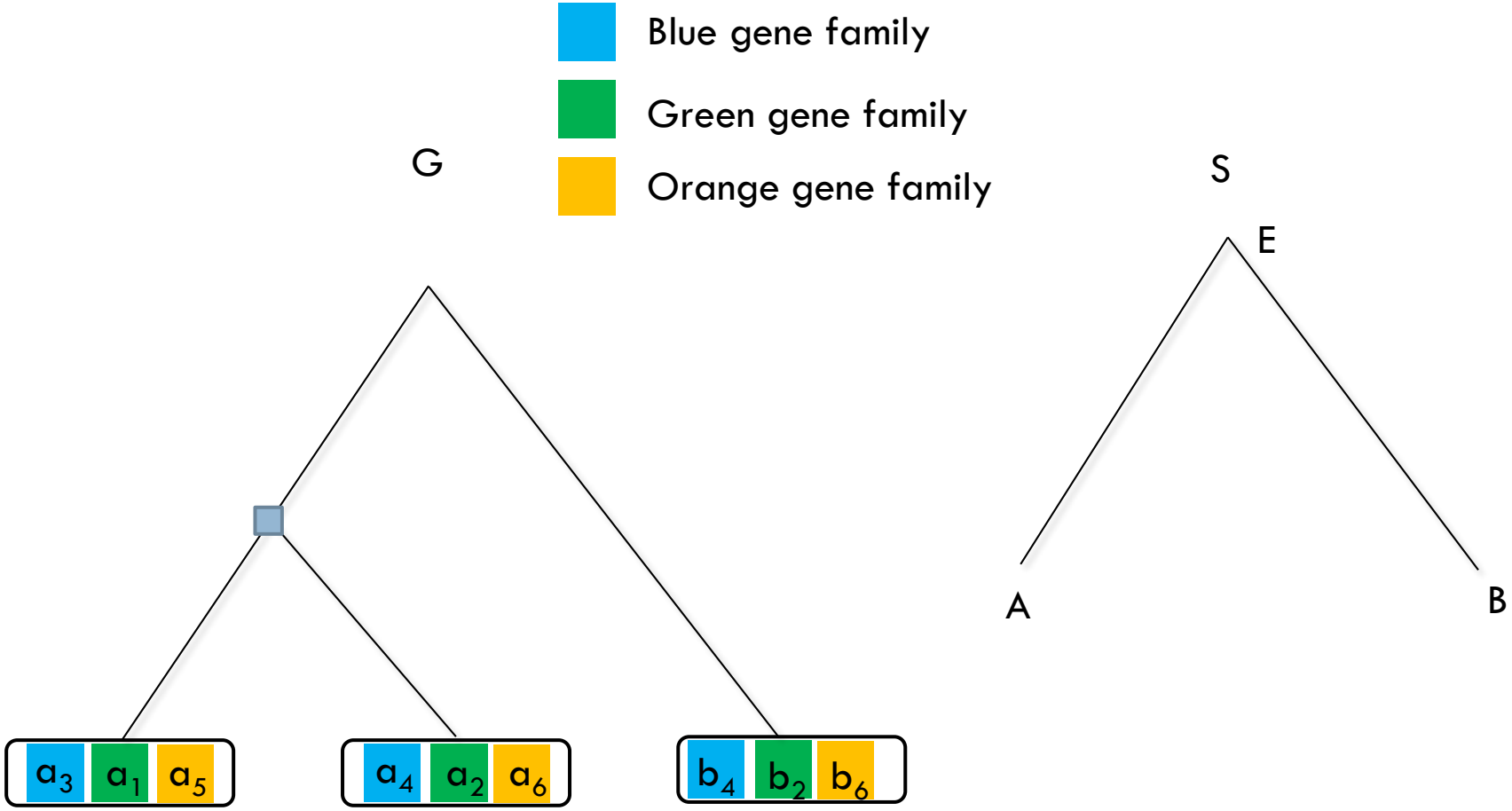

S
E
A
B


G
a_1
a_2
b_2

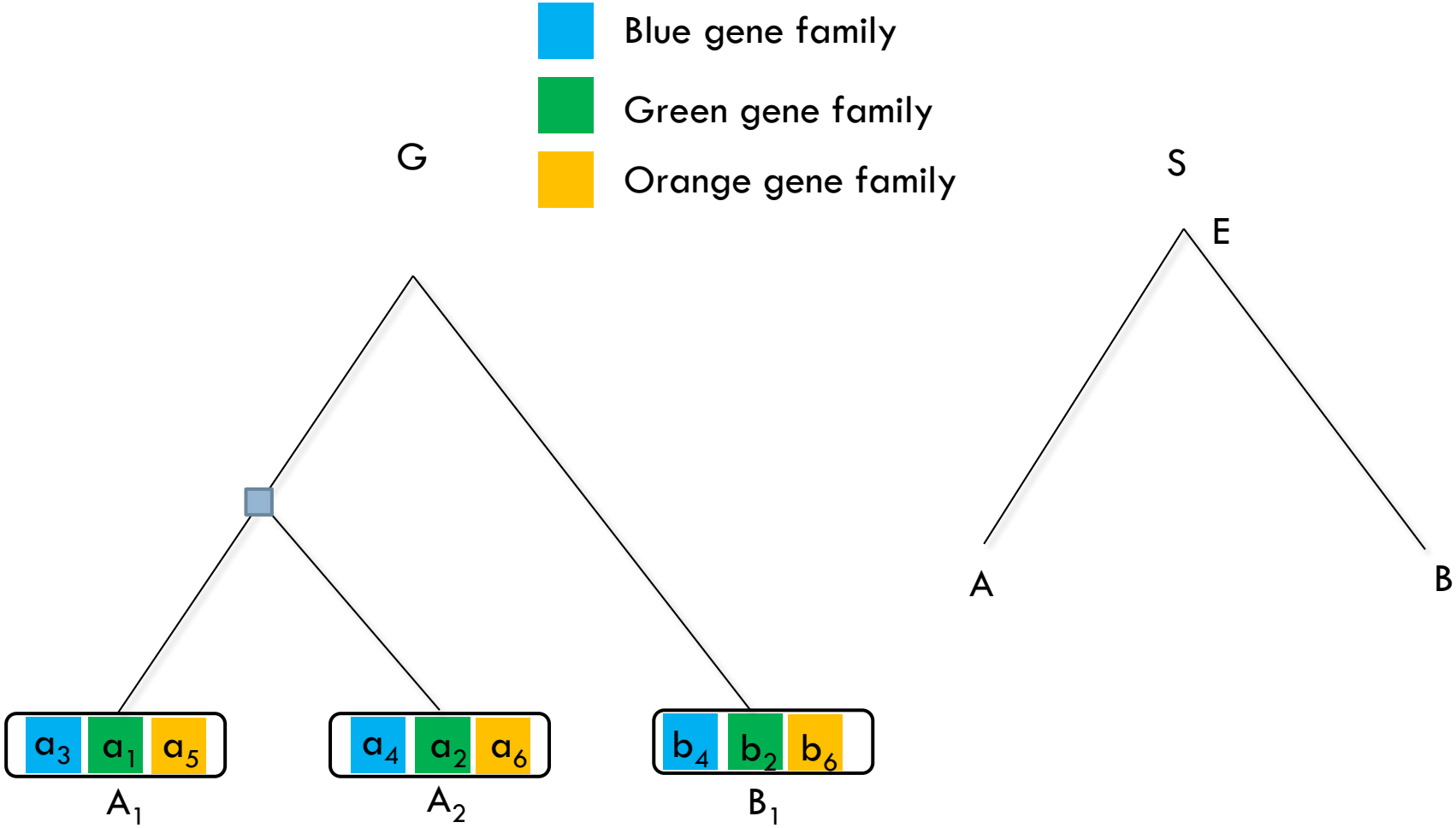

S
E
A
B


G
S
E
A
B
a_1
a_2
b_2

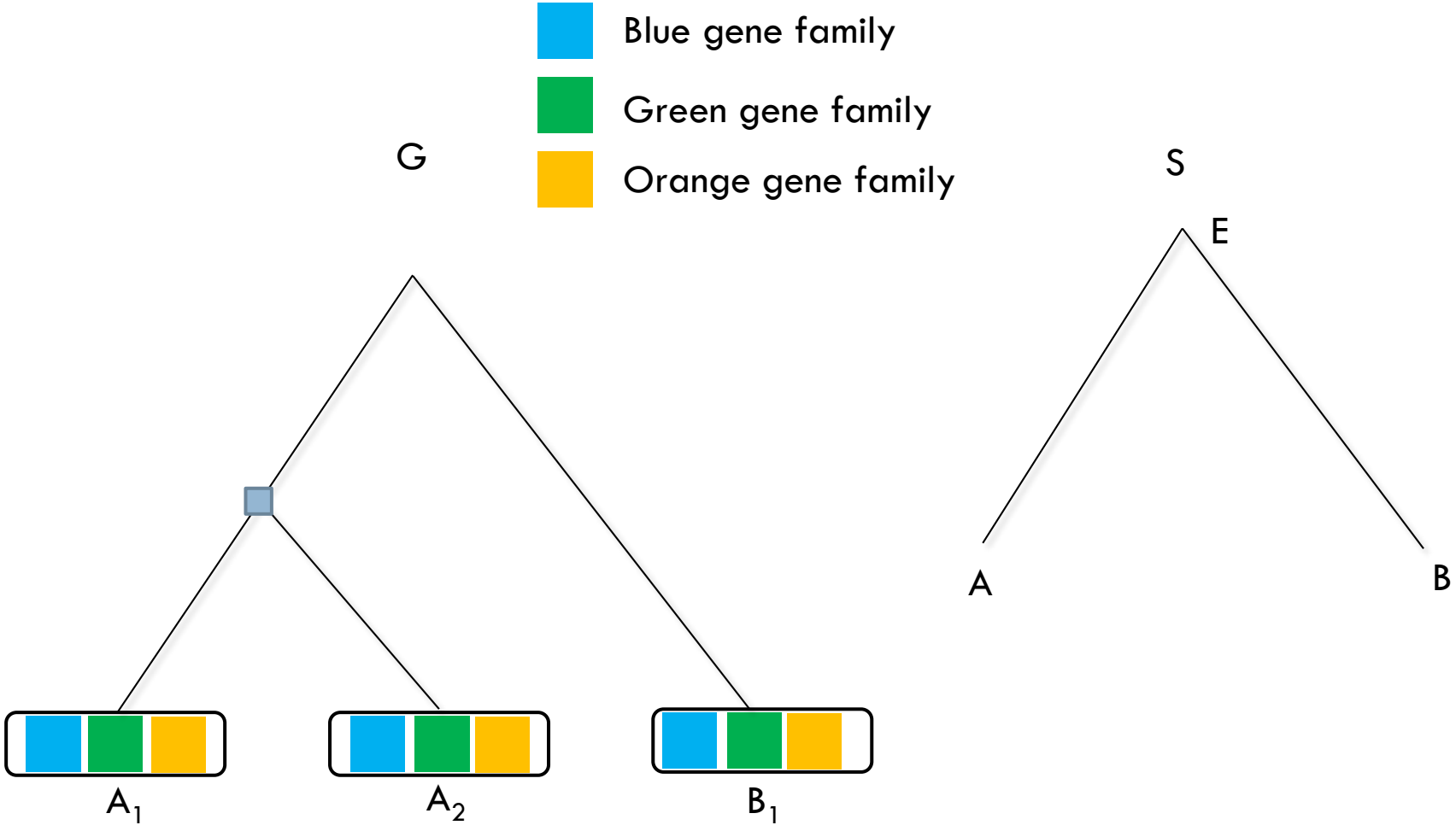

Blue gene family
Green gene family
Orange gene family
G
S
E
A
B
a_3 a_1 a_5
a_4 a_2 a_6
b_4 b_2 b_6



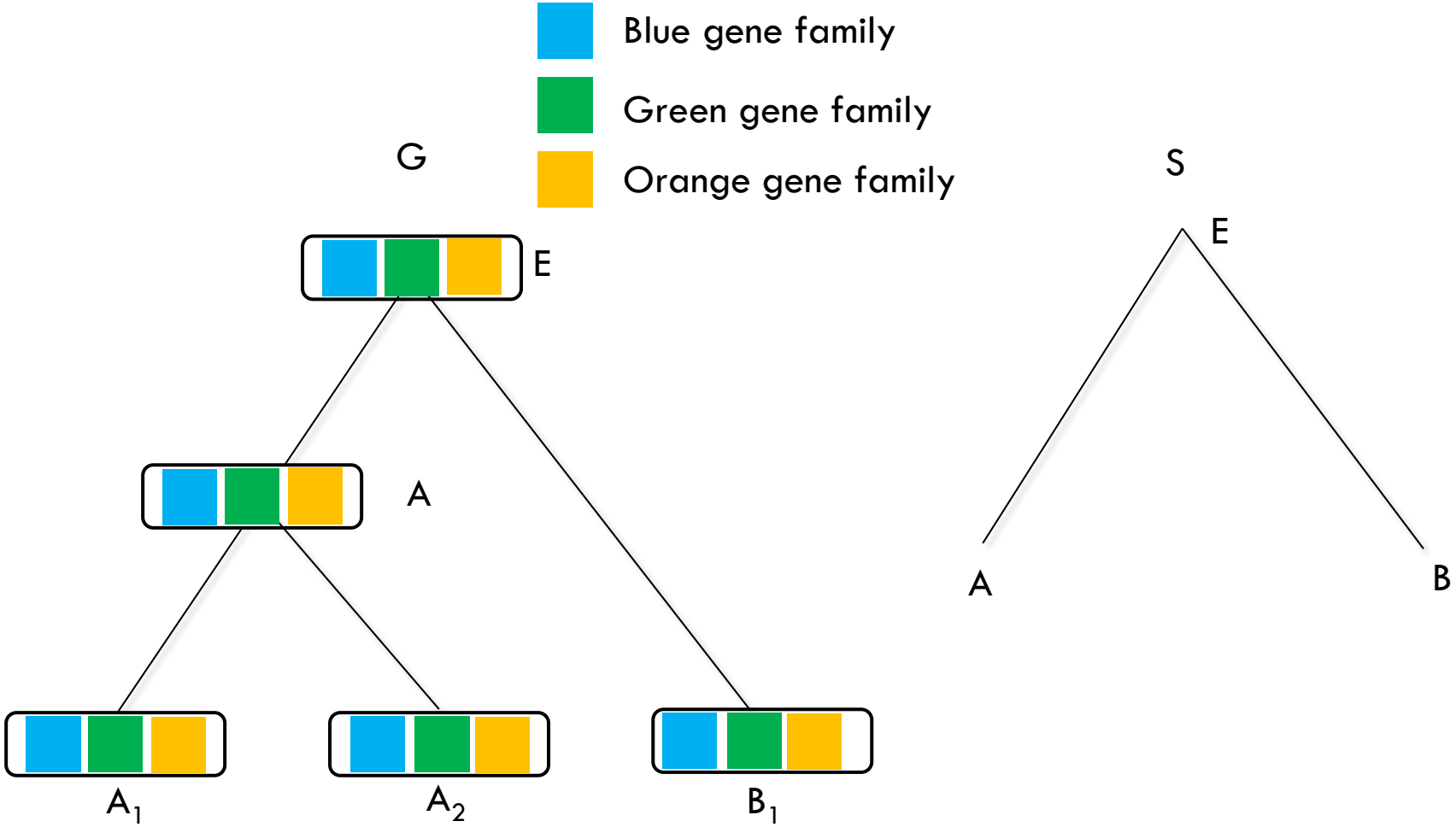
Syntenic blocks : segments with preserved gene order across species



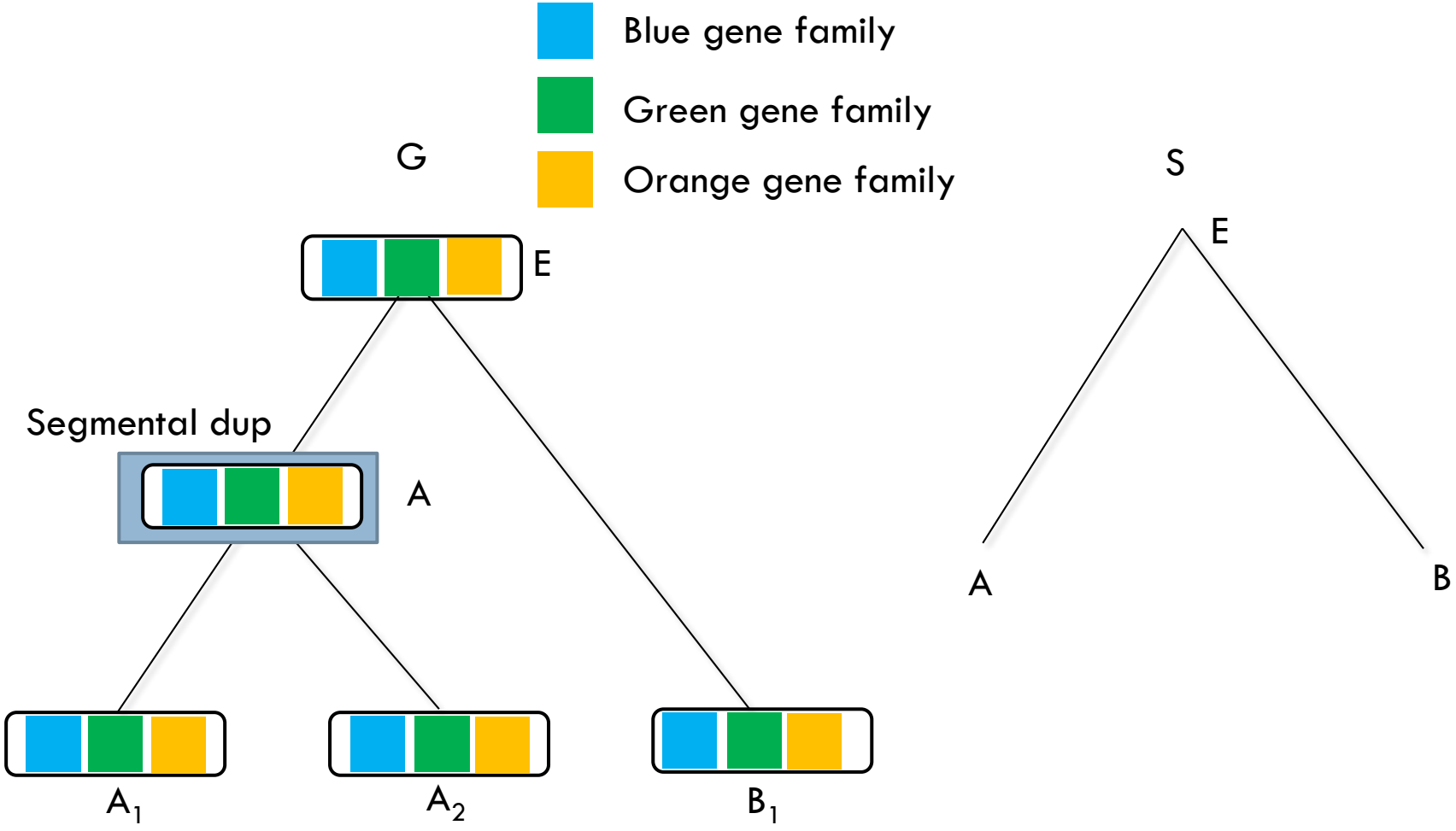
Syntenic blocks : segments with preserved gene order across species



Syntenic blocks : segments with preserved gene order across species



Syntenic blocks : segments with preserved gene order across species

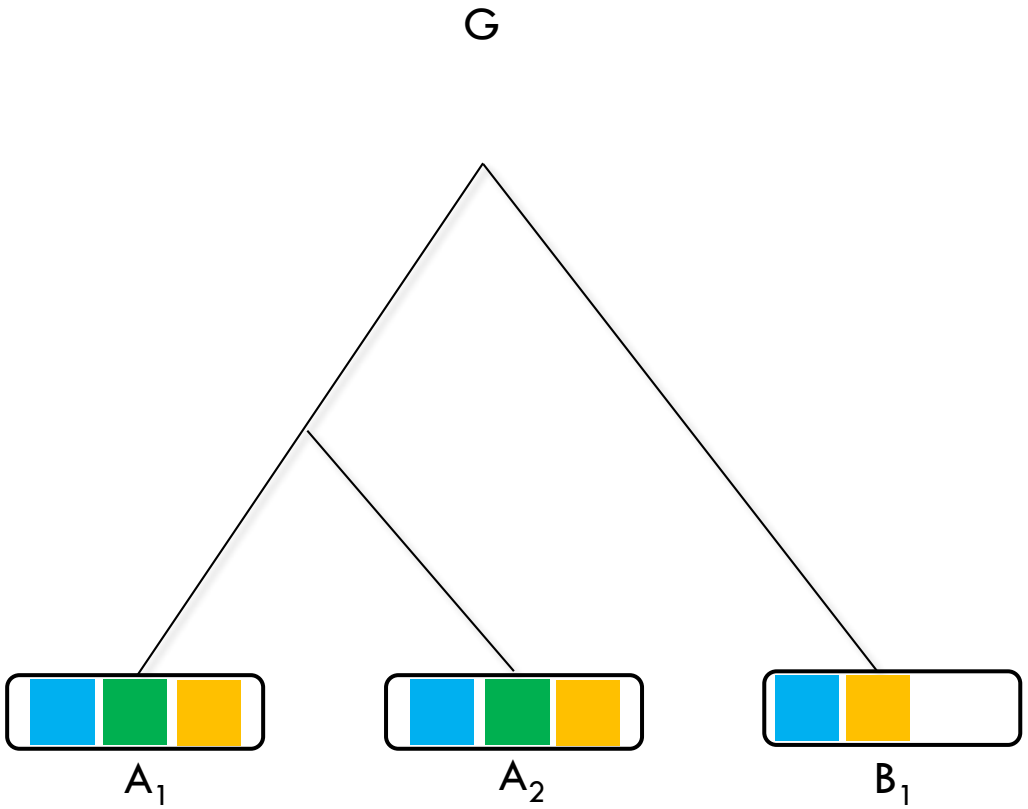


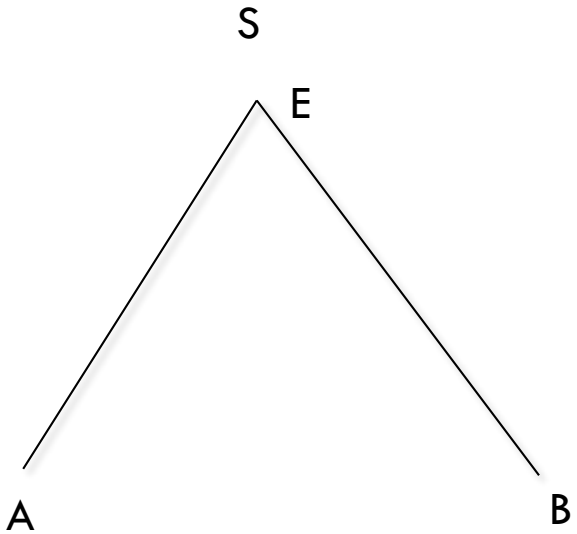
Syntenic blocks : segments with preserved gene order across species

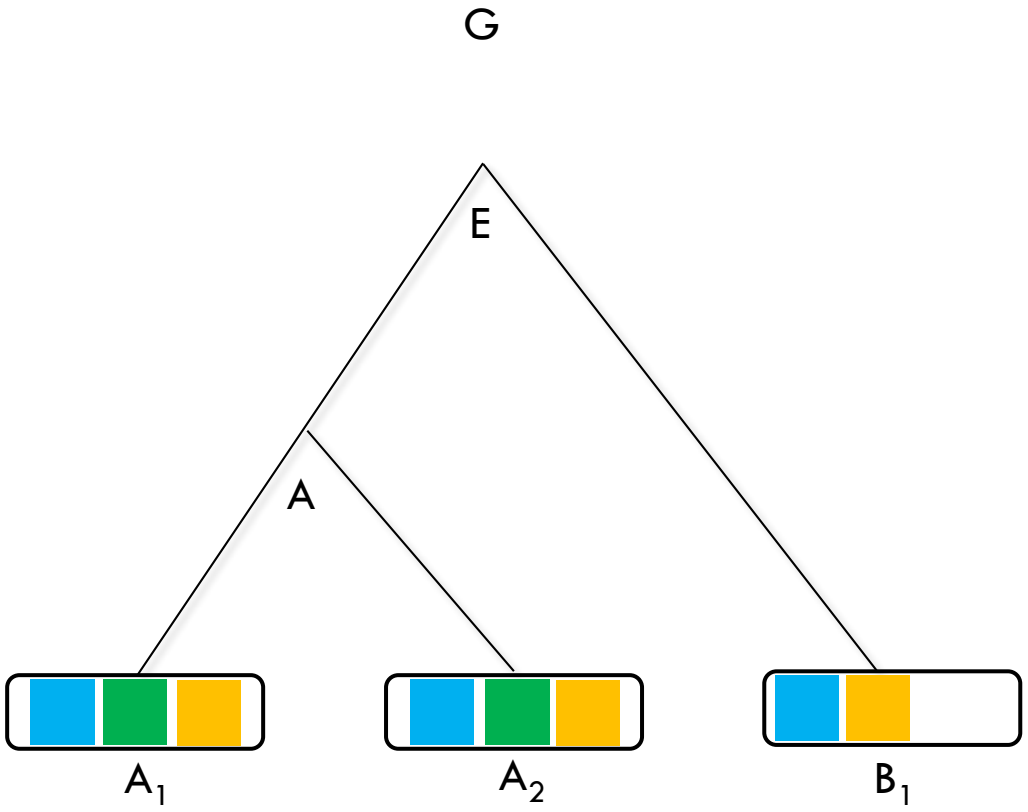
- Synteny tree reconciliation problem
  - **Input** : species tree S, a synteny tree T in which each leaf is labeled by a syntenic block
  - **Find** : an evolution of blocks across T with the minimum segmental dups + losses

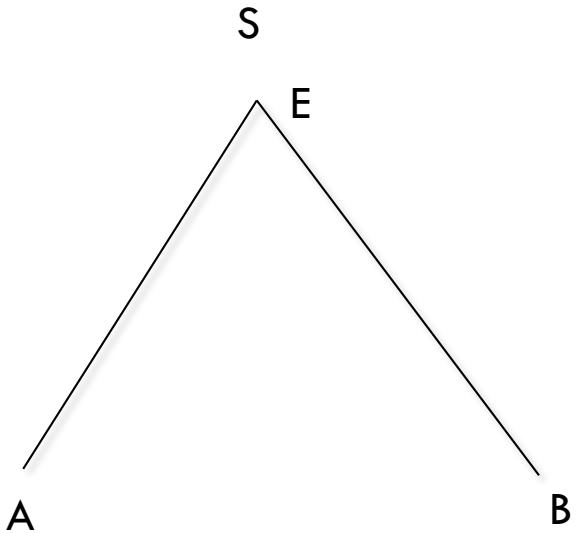
- Synteny tree reconciliation problem
  - **Input** : species tree S, a synteny tree T in which each leaf is labeled by a syntenic block
  - **Find** : an evolution of blocks across T with the minimum segmental dups + losses

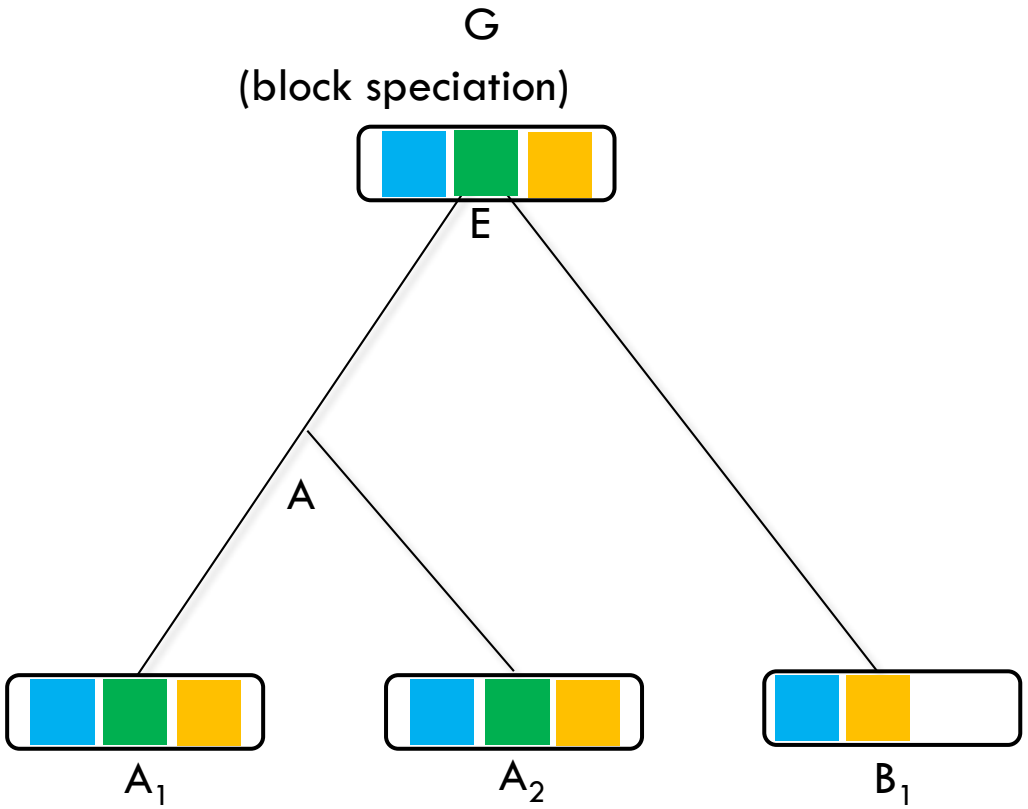
- If all leaf blocks are identical, same as classical reconciliation.
- Problem : syntenic blocks can vary in content because of losses.

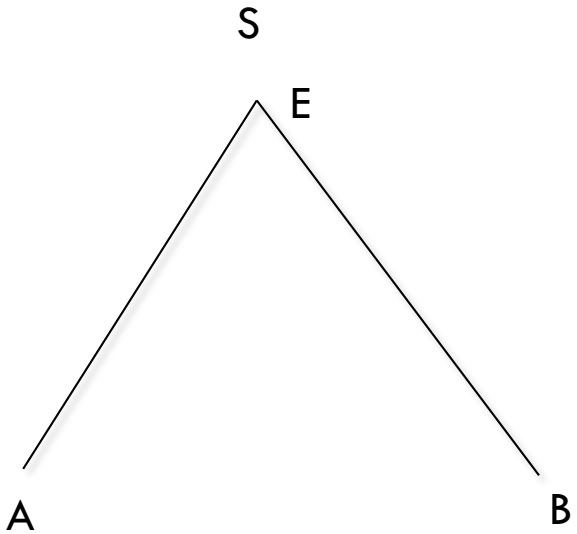

G
A_1
A_2
B_1

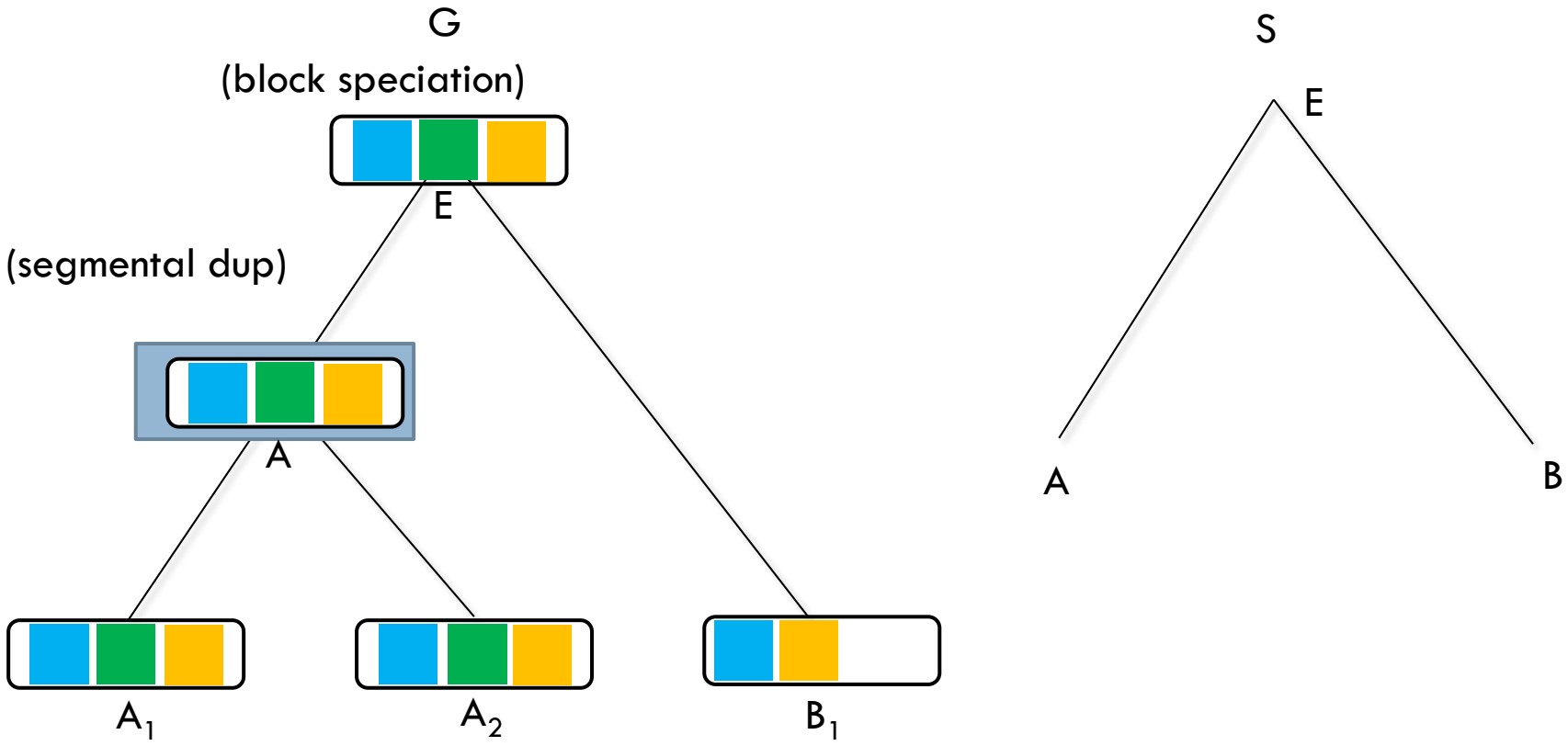

S
E
A
B

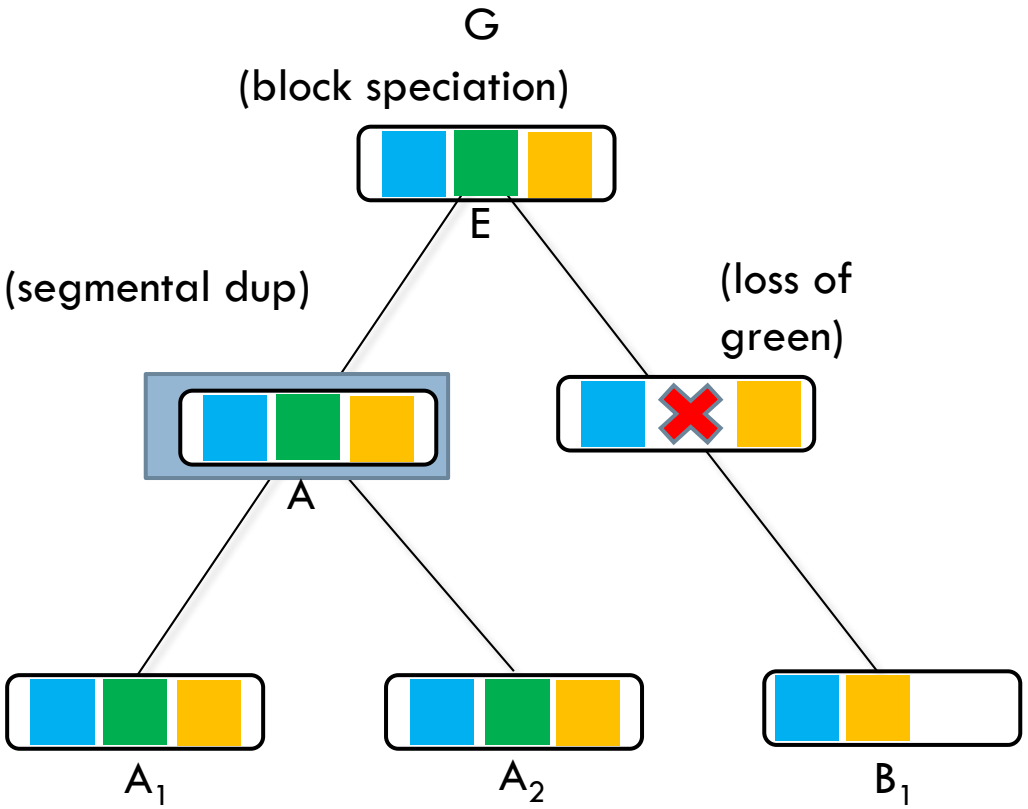

G
E
A
A_1
A_2
B_1

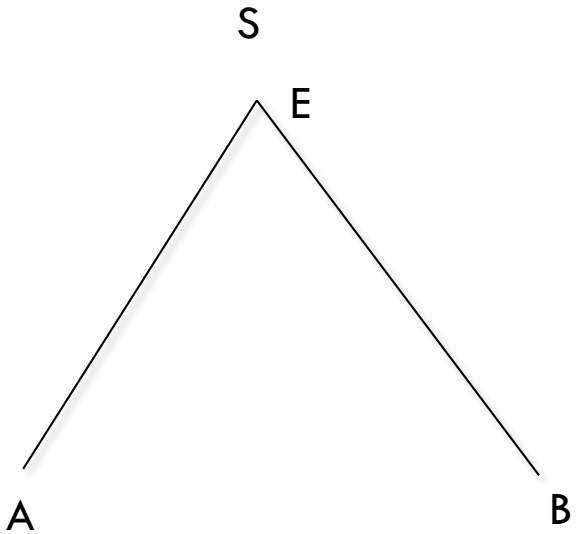

S
E
A
B


G
(block speciation)
E
A
A_1
A_2
B_1

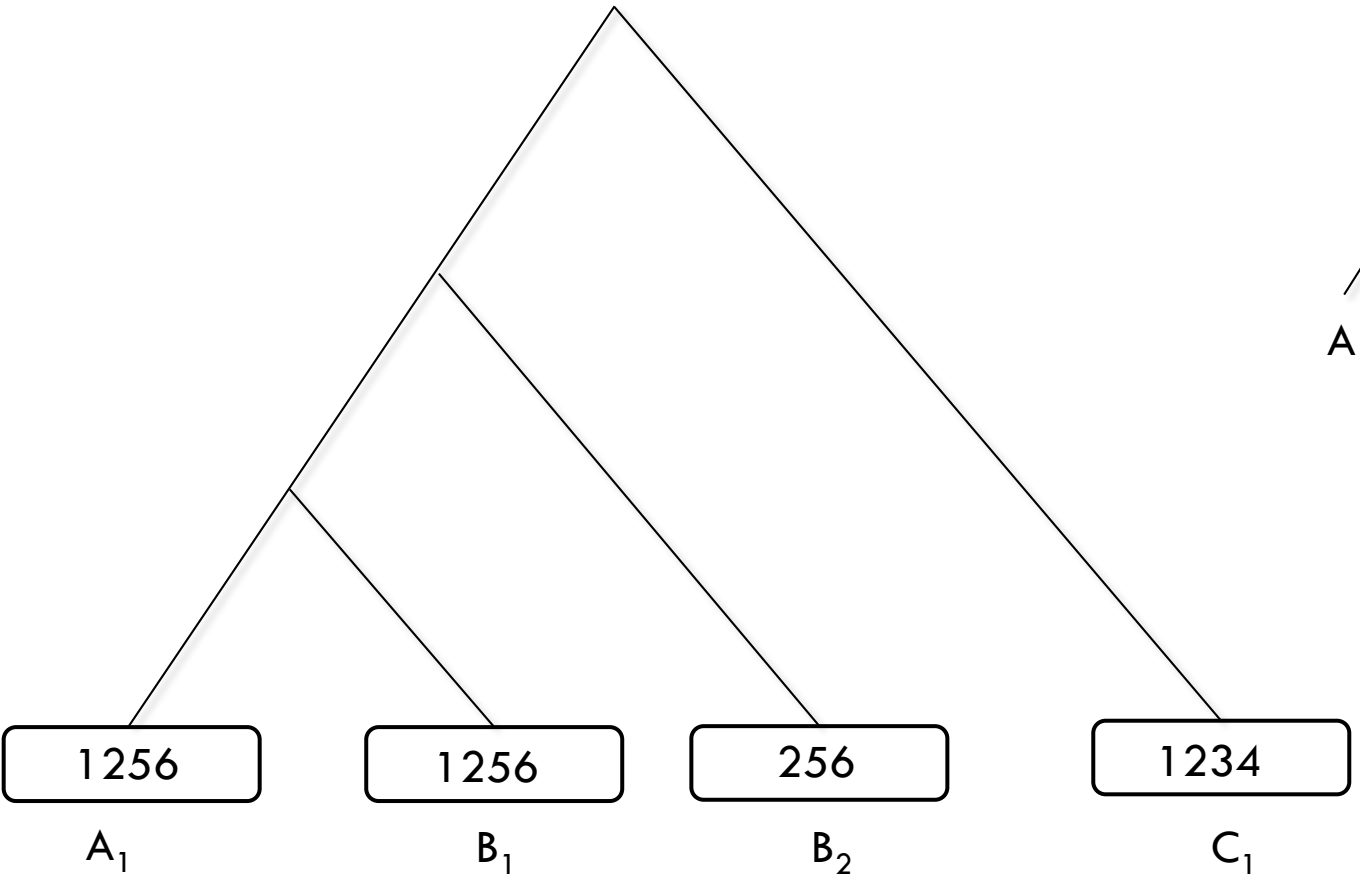

S
E
A
B

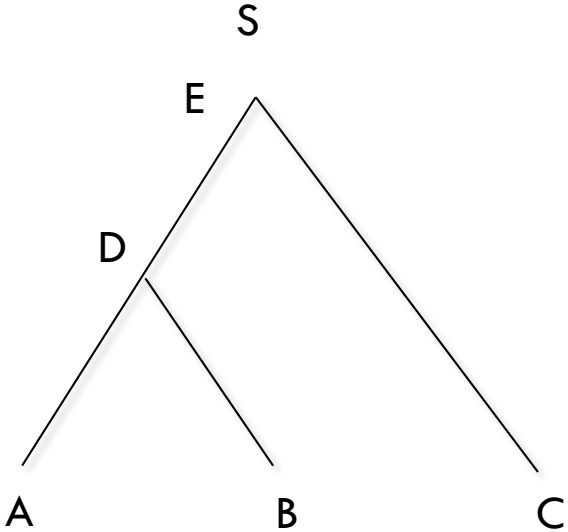

G
(block speciation)
E
(segmental dup)
A
A_1
A_2
B_1
S
E
A
B


G
(block speciation)
E
(segmental dup)
(loss of green)
A
A_1
A_2
B_1

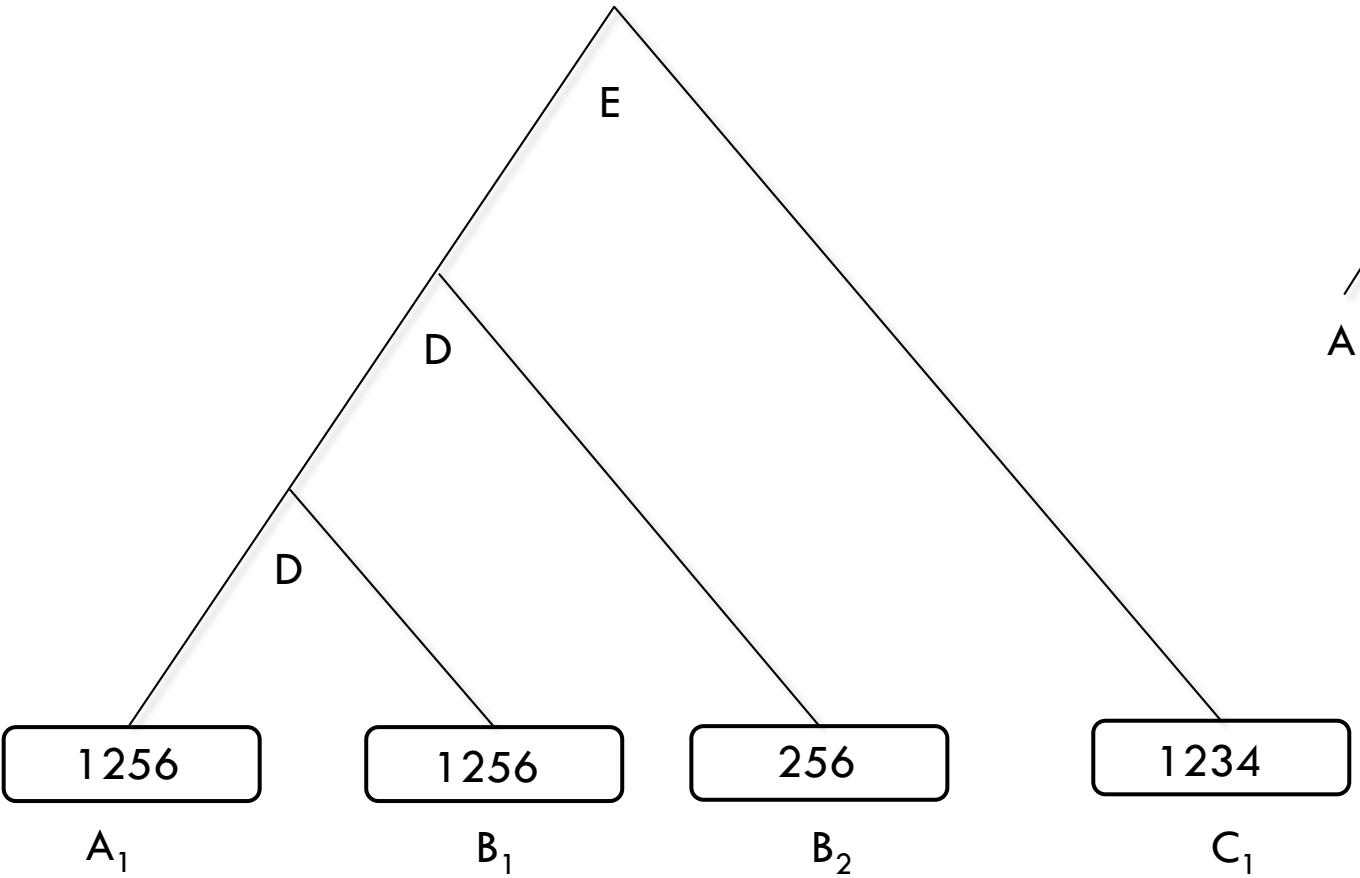

S
E
A
B

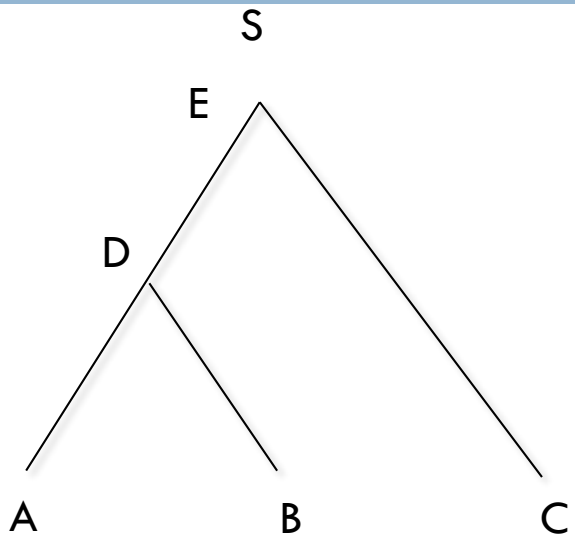
- Synteny tree reconciliation problem
  - **Input** : species tree S, a synteny tree T in which each leaf is labeled by a syntenic block
    - Syntenic block = string of characters (representing colors)
  - **Find** : an evolution of blocks across T with the minimum segmental dups + losses
    - Assign each internal node a string and a species
    - Block speciation = block transmitted to 2 descending species
    - Block dup = copy substring, paste it in a new block
    - Partial block loss = remove a substring
    - Block loss = whole block is lost



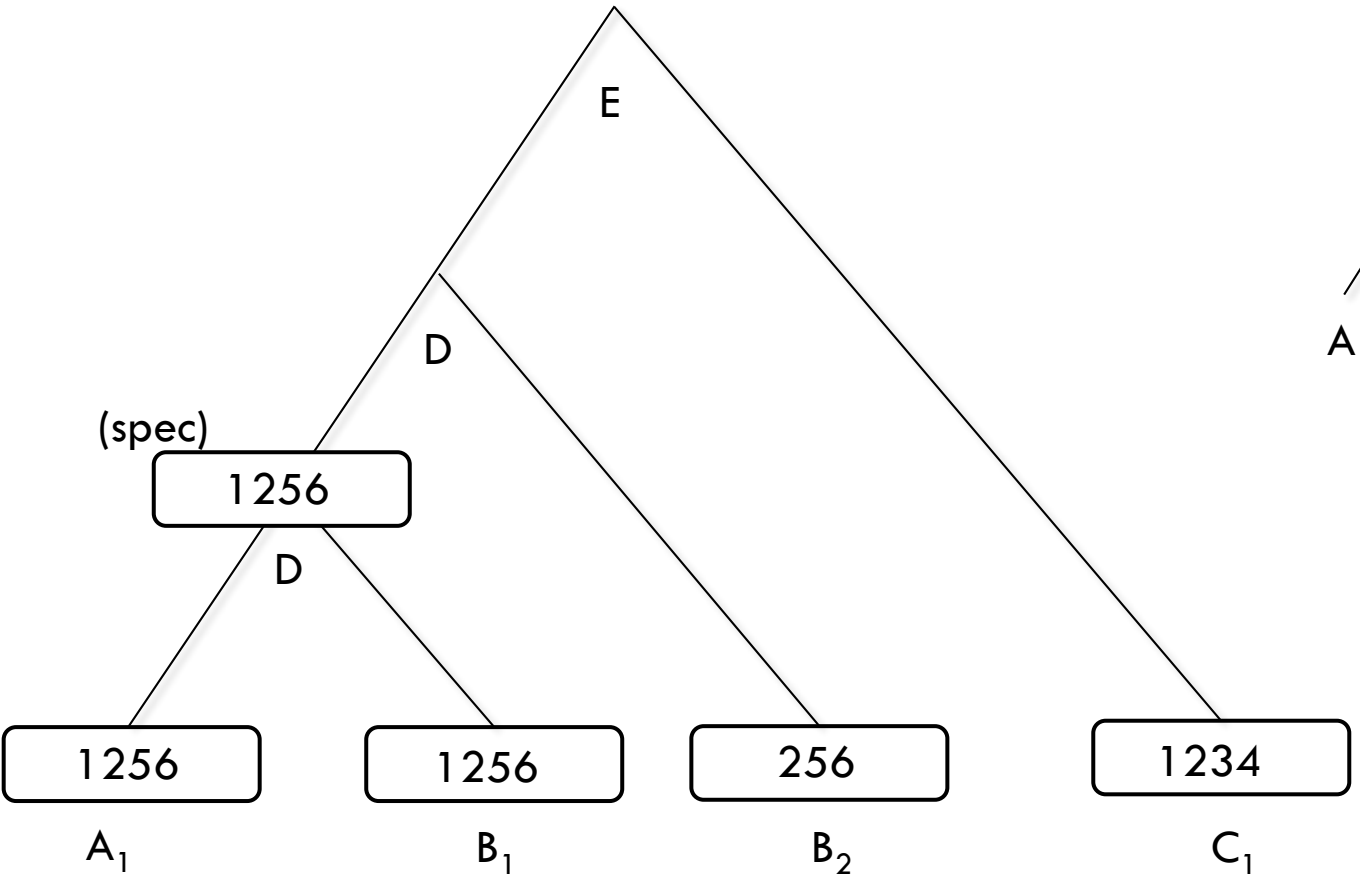


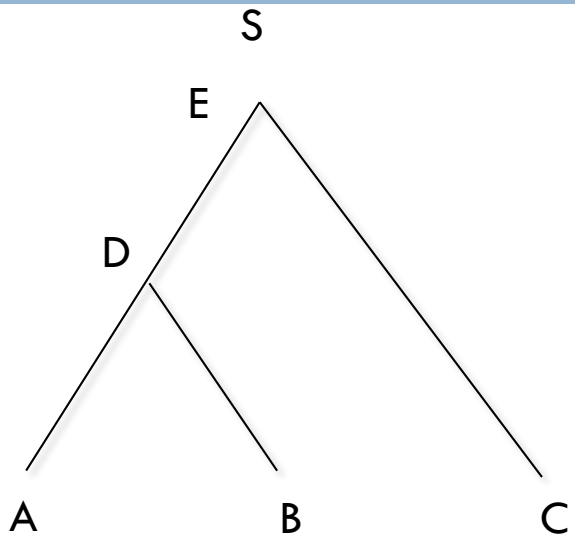
(numbers = gene family ids (instead of colors))



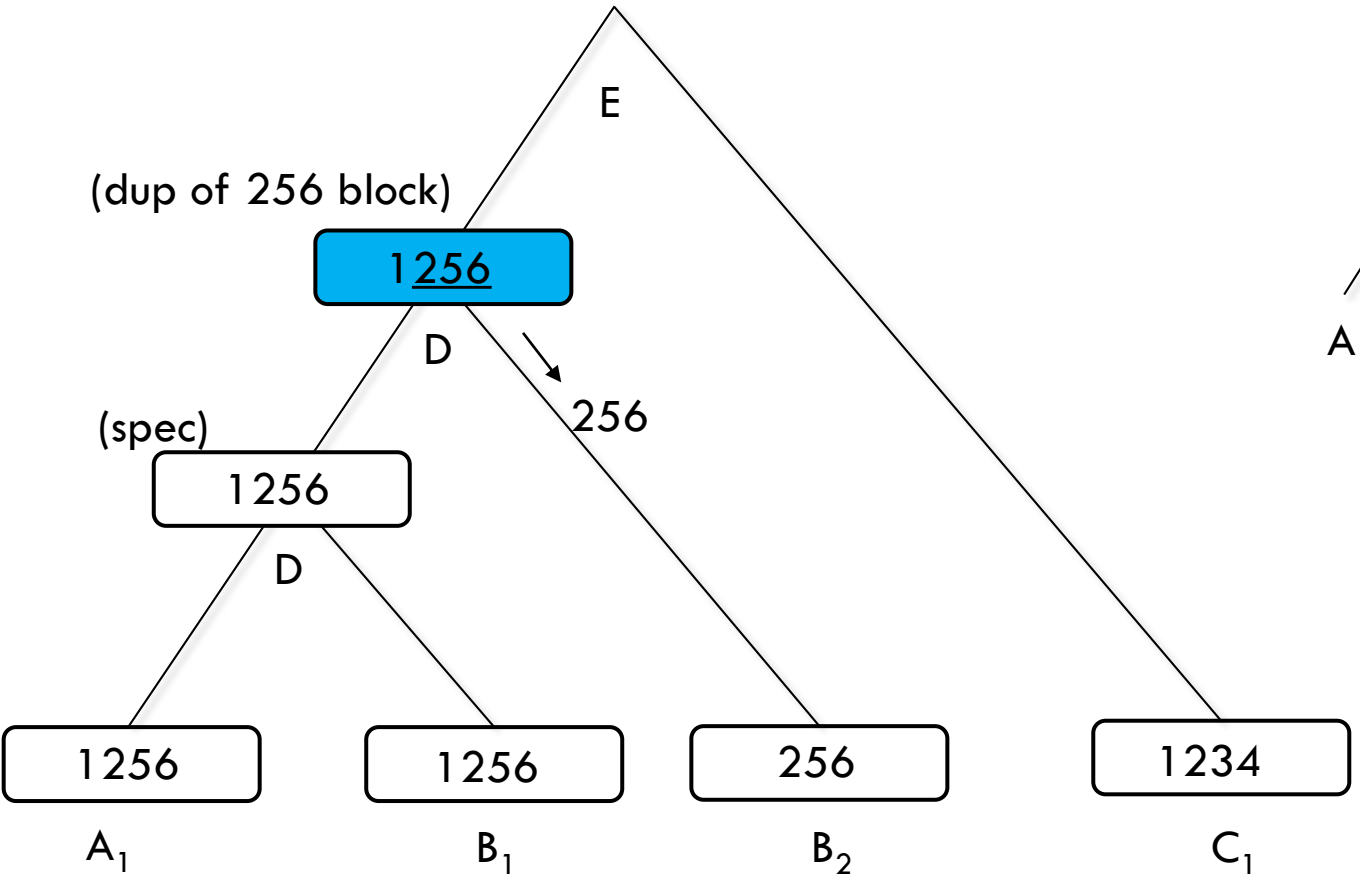


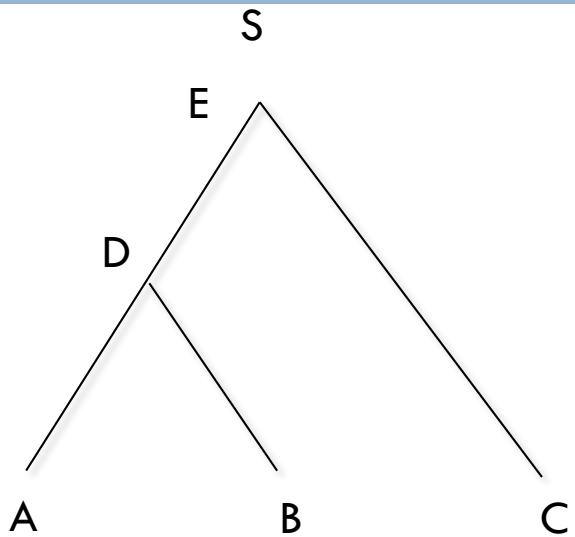
(numbers = gene family ids (instead of colors))



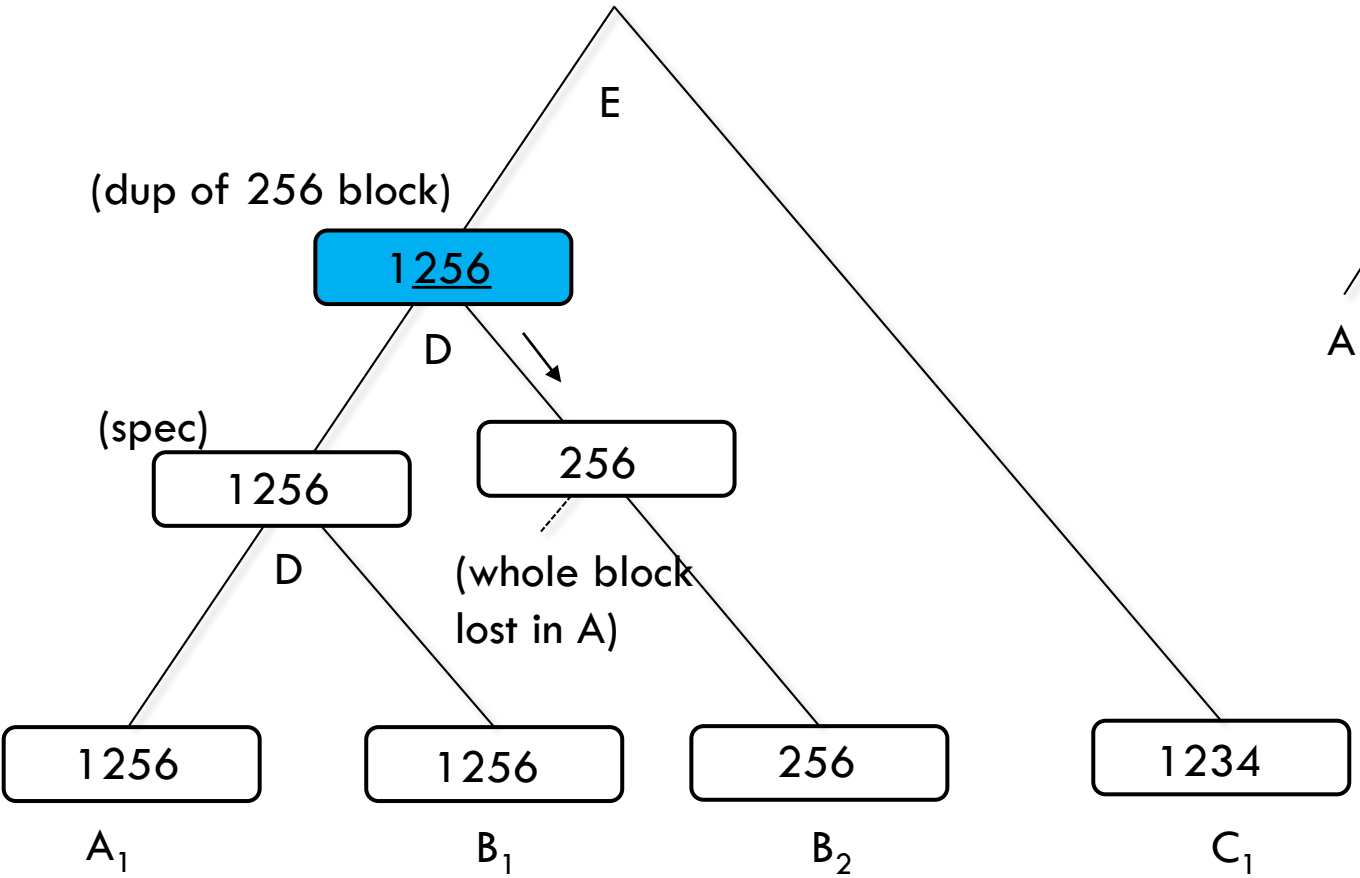


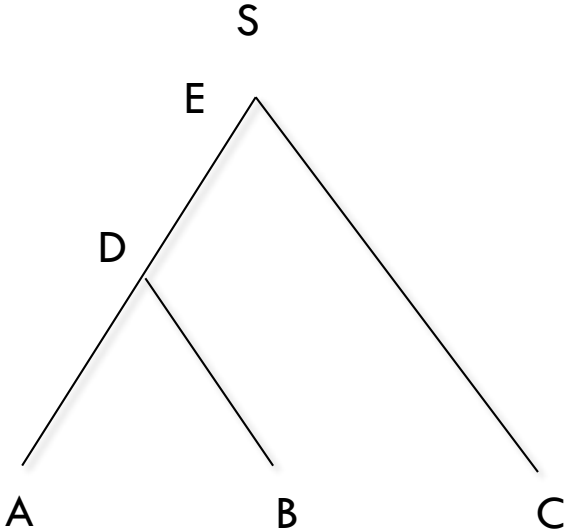
(numbers = gene family ids (instead of colors))



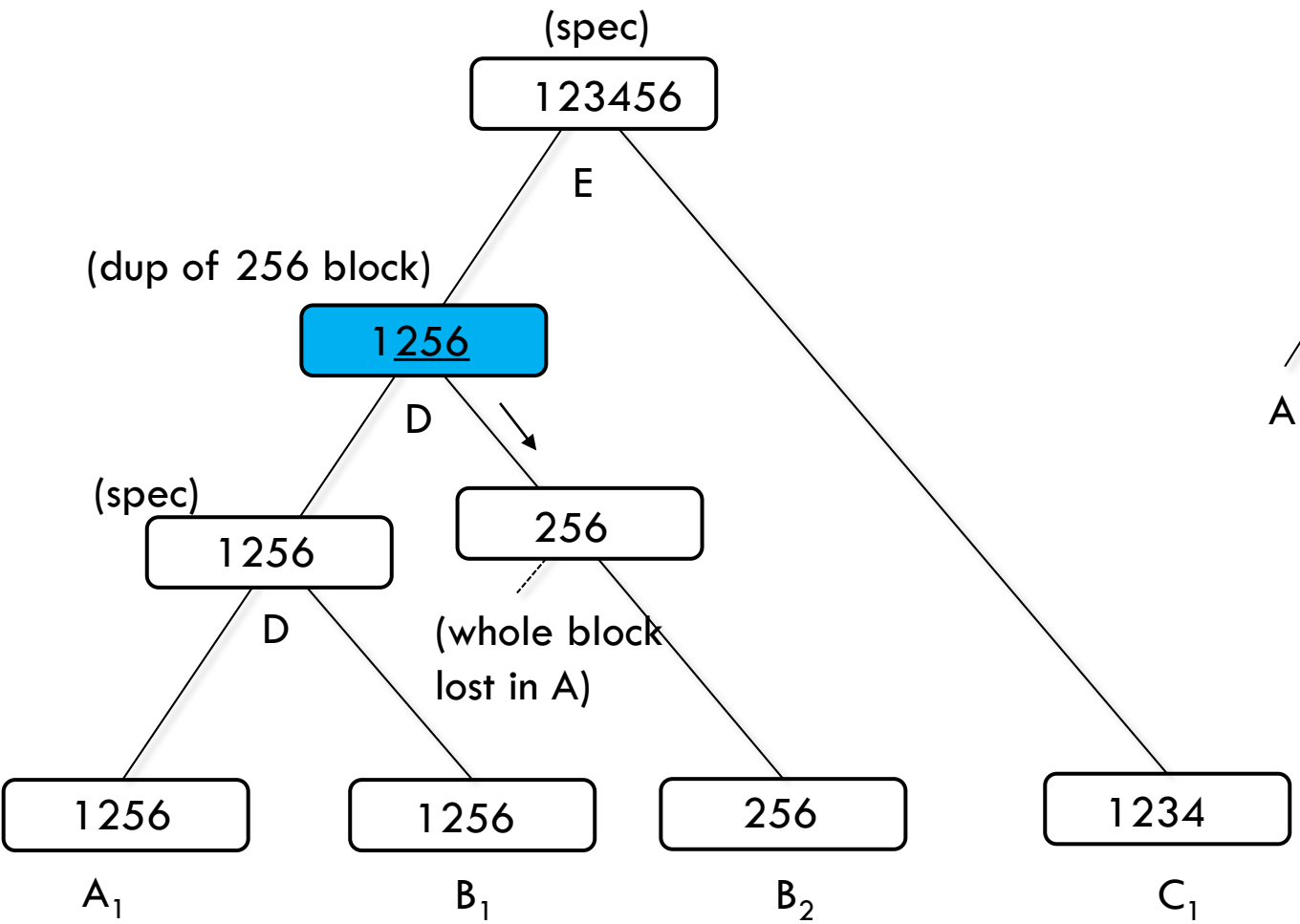


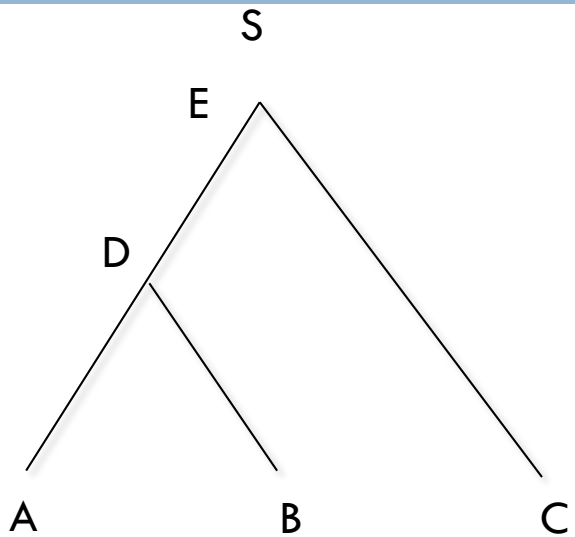
(numbers = gene family ids (instead of colors))



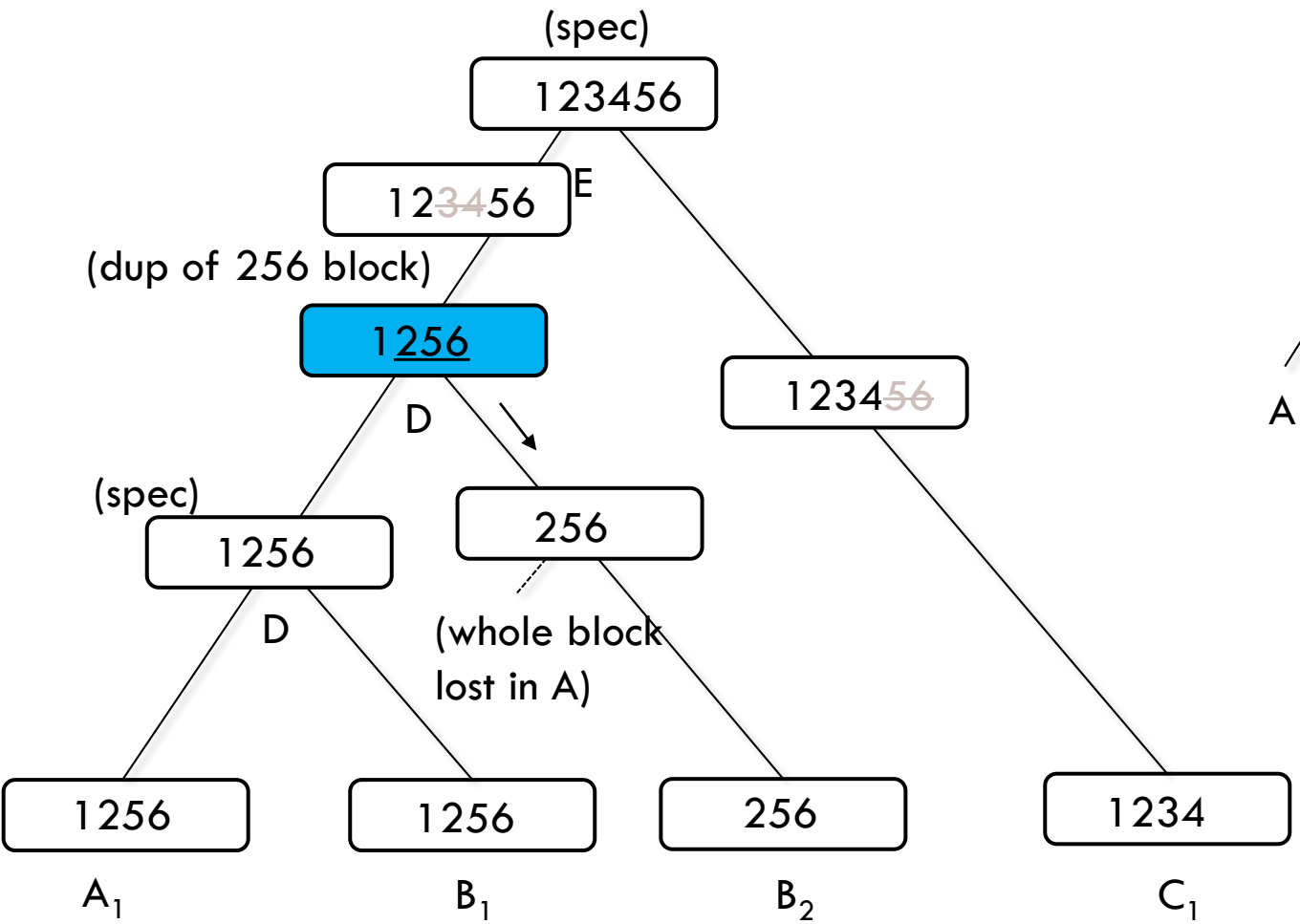


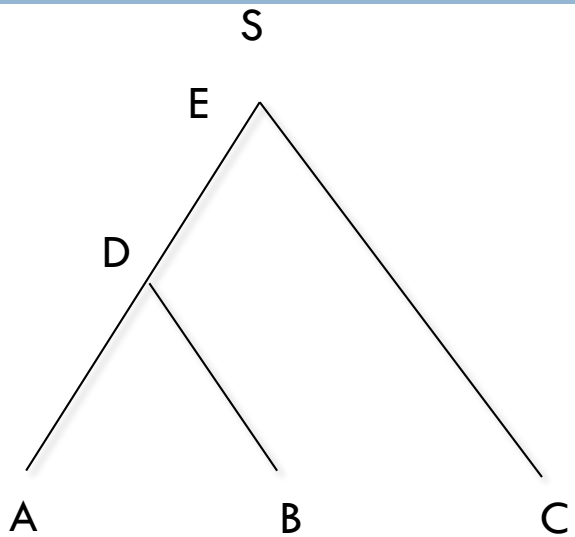
(numbers = gene family ids (instead of colors))





(numbers = gene family ids (instead of colors))





(numbers = gene family ids (instead of colors))

- Complexity of synteny tree reconciliation problem:
  - Unknown.
- Belief : if root sequence is known, feasible.

- Complexity of synteny tree reconciliation problem:
  - Unknown.
- Belief : if root sequence is known, feasible.
- Rearrangements are forbidden => the leaves give precedence constraints on the ordering of the string at the root.
  - Topological sort of leaf constraints = possible strings at the root.
  - How to choose the best ordering?

- Set version : each genome is a set of characters.
- Dup can copy any subset, loss can remove any subset.
- Can be solved in polynomial time.
  - *[Delabre, El-Mabrouk, Huber, L, Moulton, Noutahi, Sauti, AMB 2020]*
  - Bottom-up dynamic programming.

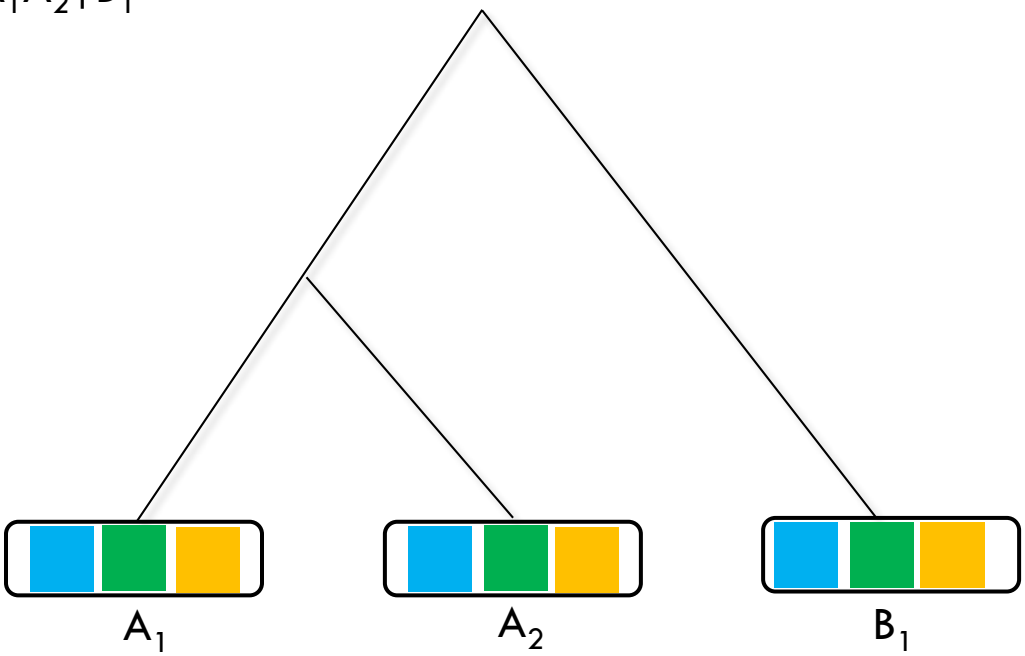
- The synteny reconciliation is just the tip of the iceberg.
- Where does the synteny tree come from?

- The synteny tree should reflect the evolution of all the genes in its blocks.
  - Each gene family has its own tree.
  - Each tree = an 'opinion' on how the blocks evolved.
  - If each tree is identical, synteny tree is obvious.
  - If not, the 'opinions' should at least be **compatible**.

Incompatible histories

How did the blocks A_1, A_2, B_1 evolve?

Green family says : $A_1A_2|B_1$

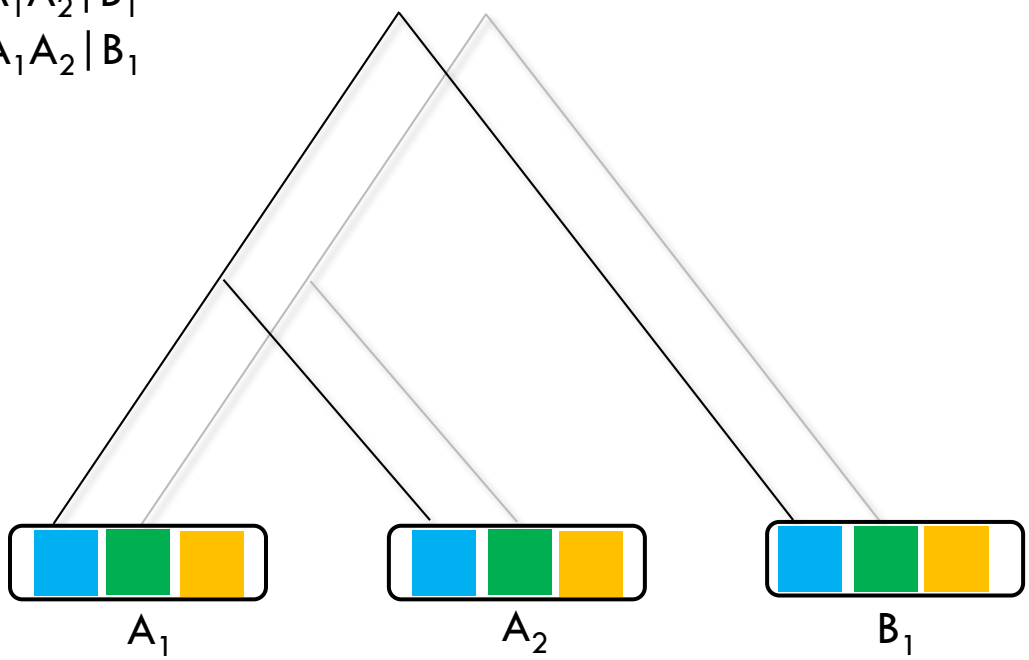


Incompatible histories

How did the blocks A_1, A_2, B_1 evolve?

Green family says : $A_1A_2|B_1$

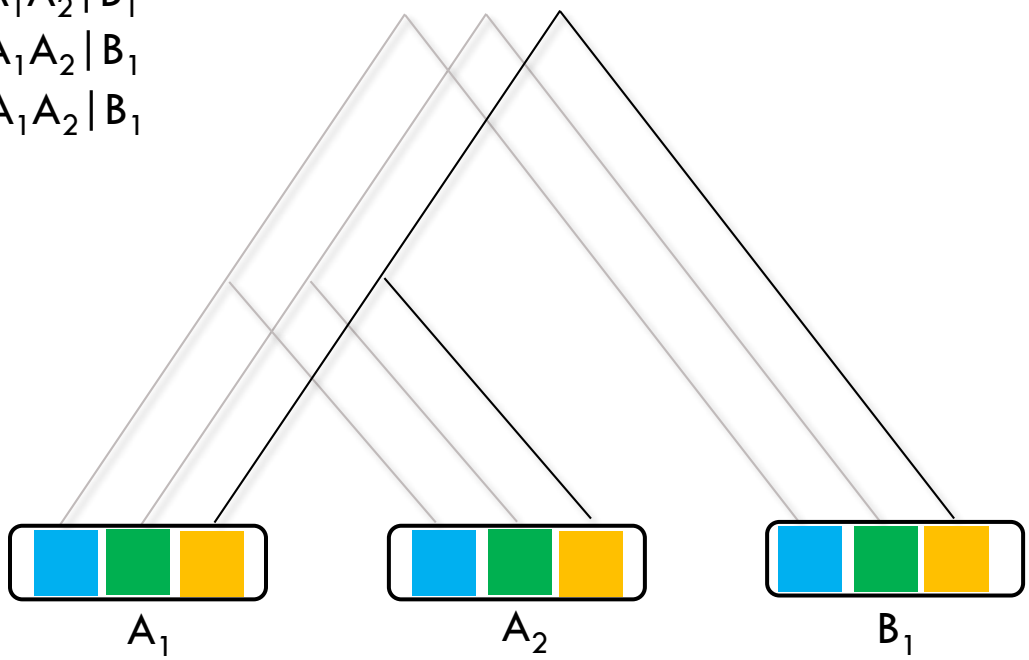
Blue family says : $A_1A_2|B_1$



Incompatible histories

How did the blocks A_1, A_2, B_1 evolve?

Green family says : $A_1A_2|B_1$
Blue family says : $A_1A_2|B_1$
Orange family says : $A_1A_2|B_1$

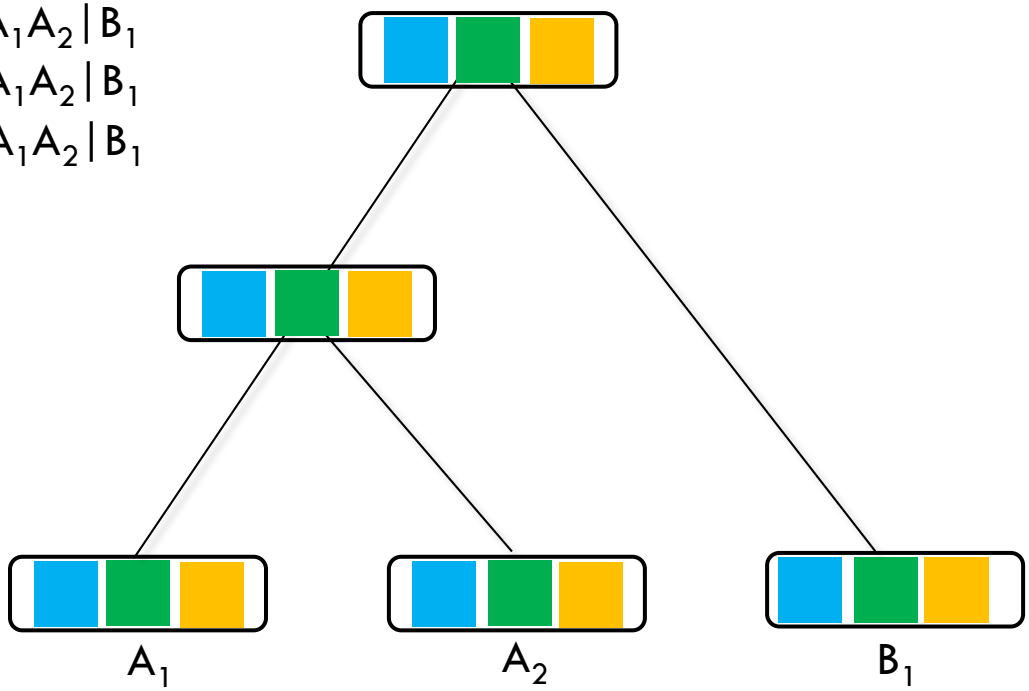


All trees agree => Good, synteny tree is obvious.

Incompatible histories

How did the blocks A_1, A_2, B_1 evolve?

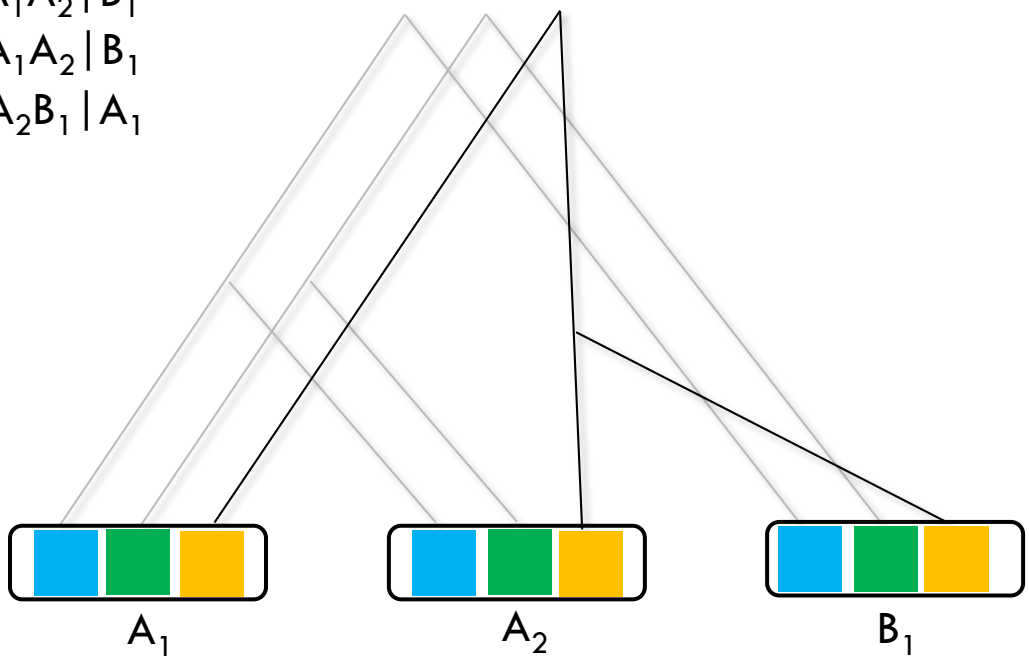
Green family says : $A_1A_2|B_1$
Blue family says : $A_1A_2|B_1$
Orange family says : $A_1A_2|B_1$



All trees agree => Good, synteny tree is obvious.

Incompatible histories

Green family says : $A_1A_2|B_1$
Blue family says : $A_1A_2|B_1$
Orange family says : $A_2B_1|A_1$



Discordant topologies => Blocks can't have evolved together.

Incompatible histories

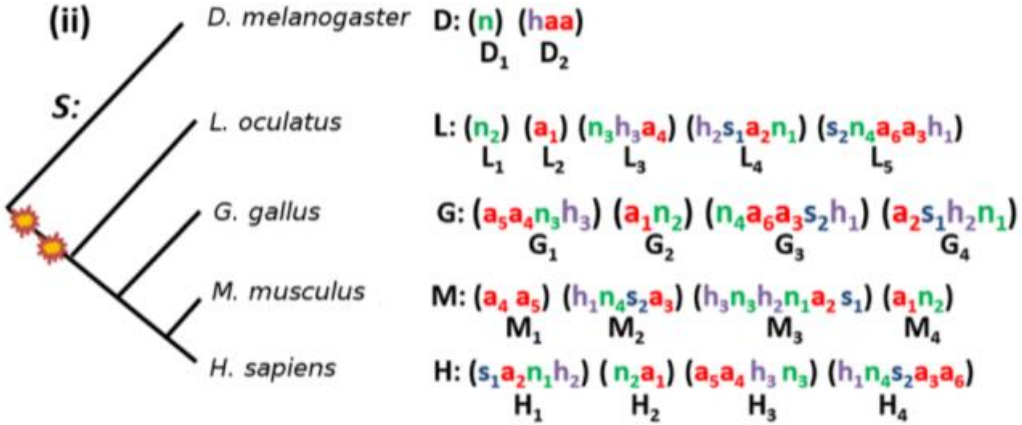
- If each synteny has exactly k genes, we have k trees with leafset X = leaf labels = blocks, all distinct
- If the k trees are identical, easy to reconcile.
- If not, find **some minimal way** to edit the trees so that they are identical + **minimize reconciliation cost**.
  - No clear formulation known.

Proof-of-concept : opioids family

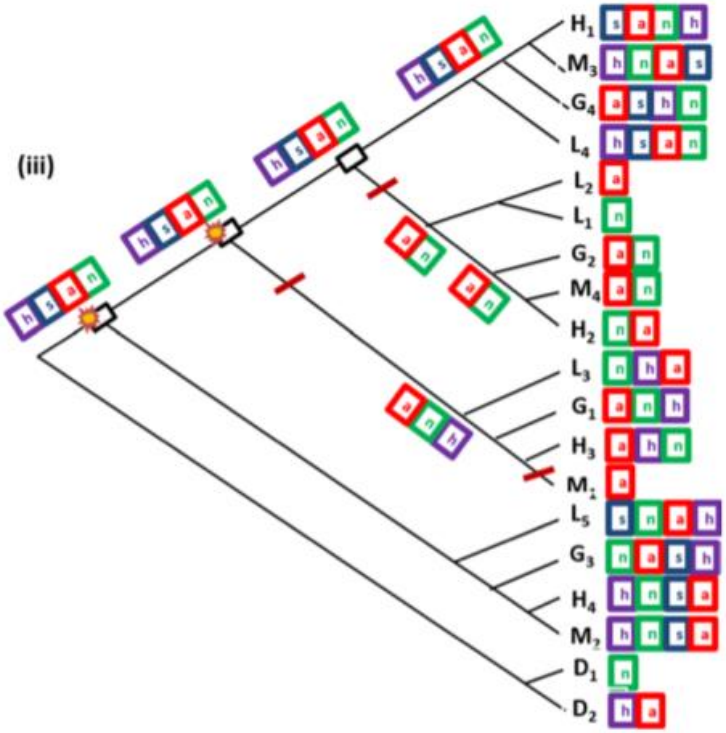
(i)

OPR	NKAIN	STMN	SRC-B
OPRM1 (a_1)	NKAIN1 (n_1)	STMN1 (s_1)	HCK (h_1)
OPRD1 (a_2)	NKAIN2 (n_2)	STMN3 (s_2)	LCK (h_2)
OPRL1 (a_3)	NKAIN3 (n_3)		LYN (h_3)
OPRK1 (a_4)	NKAIN4 (n_4)		
NPBWR1 (a5)			
NPBWR2 (a6)			

(ii)



(iii)



Conclusion

- Modeling challenges
  - Combine both ME and synteny views
  - Infer ME dups + losses using synteny information
  - Integrate segmental losses + transfers + …
  - Integrate order-changing rearrangements

  - Also, simulate good multi-family evolutionary scenarios

Conclusion

- Algorithmic challenges
  - Scalable algorithm for ME + losses inference
  - Synteny tree reconciliation and variants
  - Construct a synteny tree that agrees the most with the gene trees in the blocks

Thank you