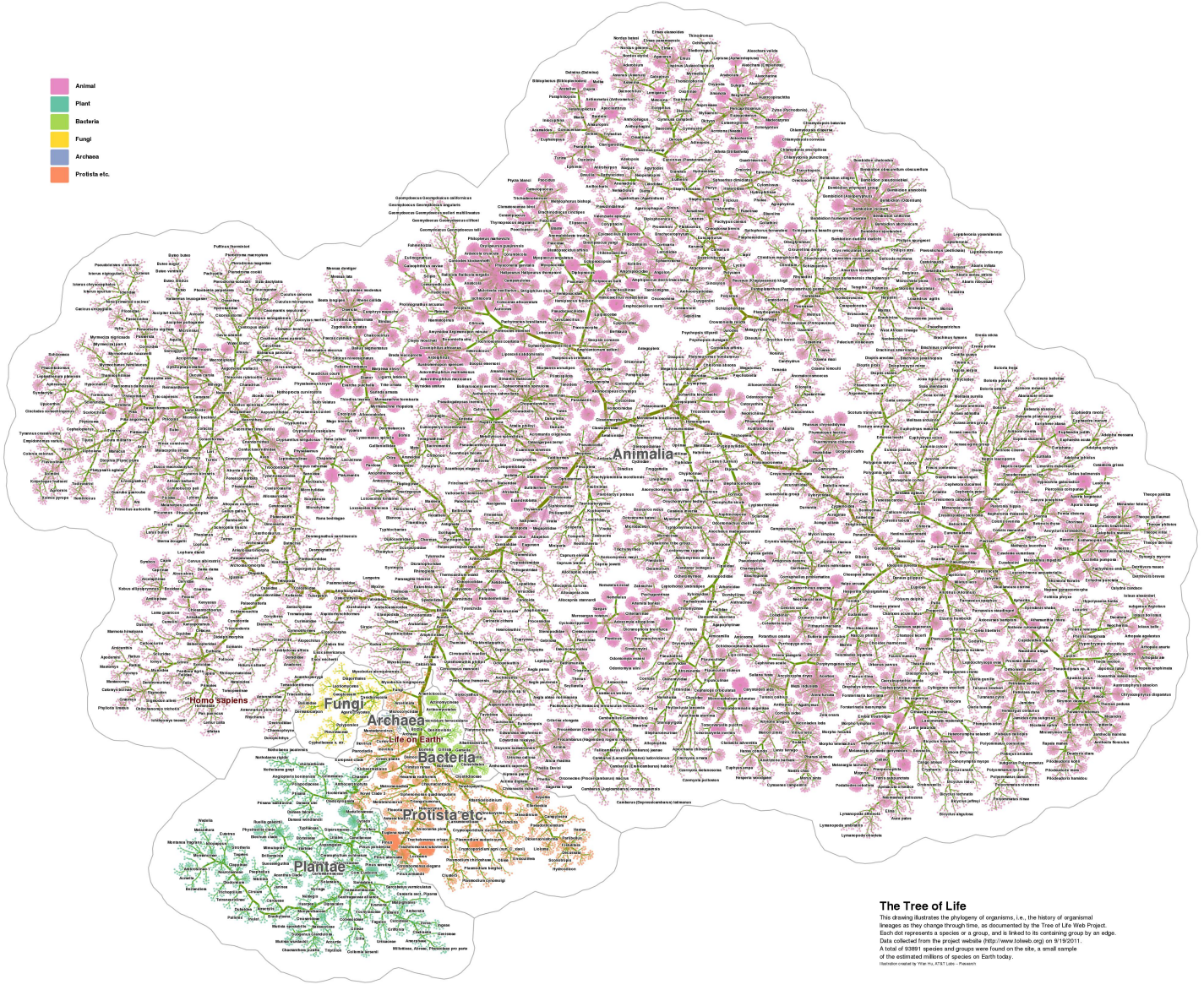
Combinatorial Optimization in Bioinfo

Lecture 6 – Comparative genomics

Yann Ponty

AMIBio Team
CNRS & École Polytechnique

The tree of life



Cool story bro. What next?

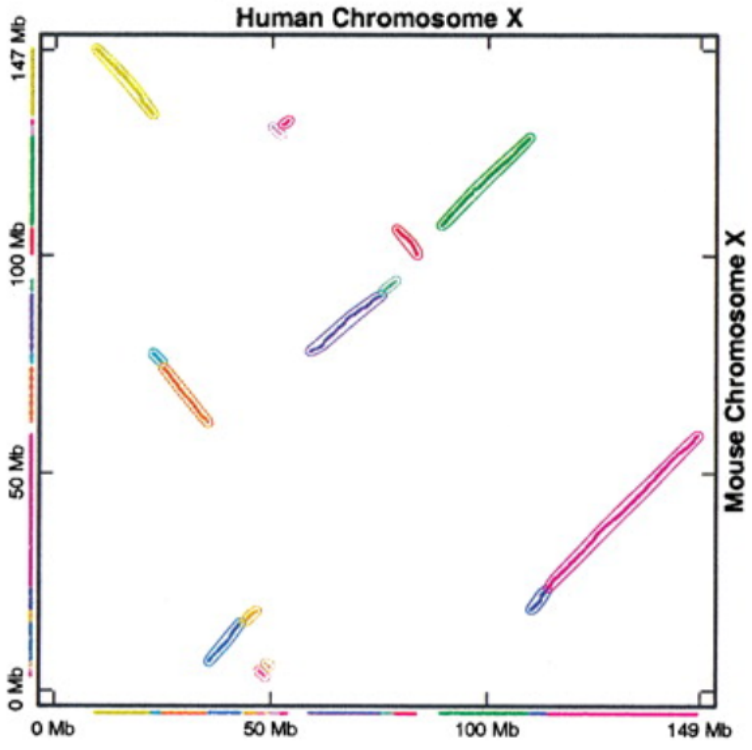
Comparative genomics studies (epi)genomics features of current (extant) species, to gain insight into their evolution and ancestral traits.

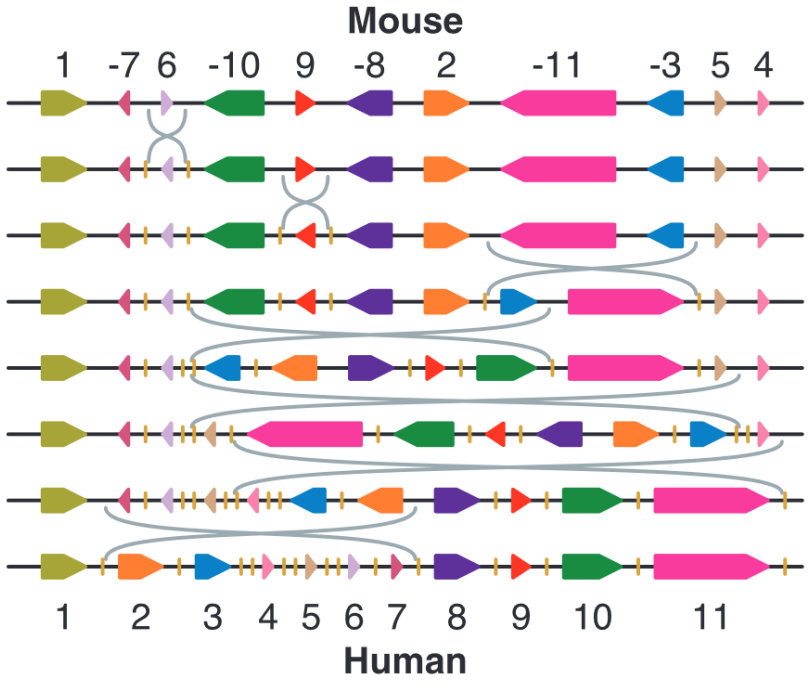
Some examples:

- Accurate inter-species distances by integrating genome-scale organization (genome rearrangements);
- Inference of ancestral features (genomic adjacencies);
- Reconciliations of species and gene trees. . .

Parsimony as a common framework (but not exclusively!)

At **genomic scale** + **longer periods of time**, classic sequence homology becomes insufficient to infer elapsed time and precise correspondences.





[Pevzner and Tesler, Genome Research 2003]

How many **inversions** required to turn a Mouse into a Human?
sort a signed permutation by reversals?

SORTING BY REVERSALS **problem**

Input: *Signed* permutation π

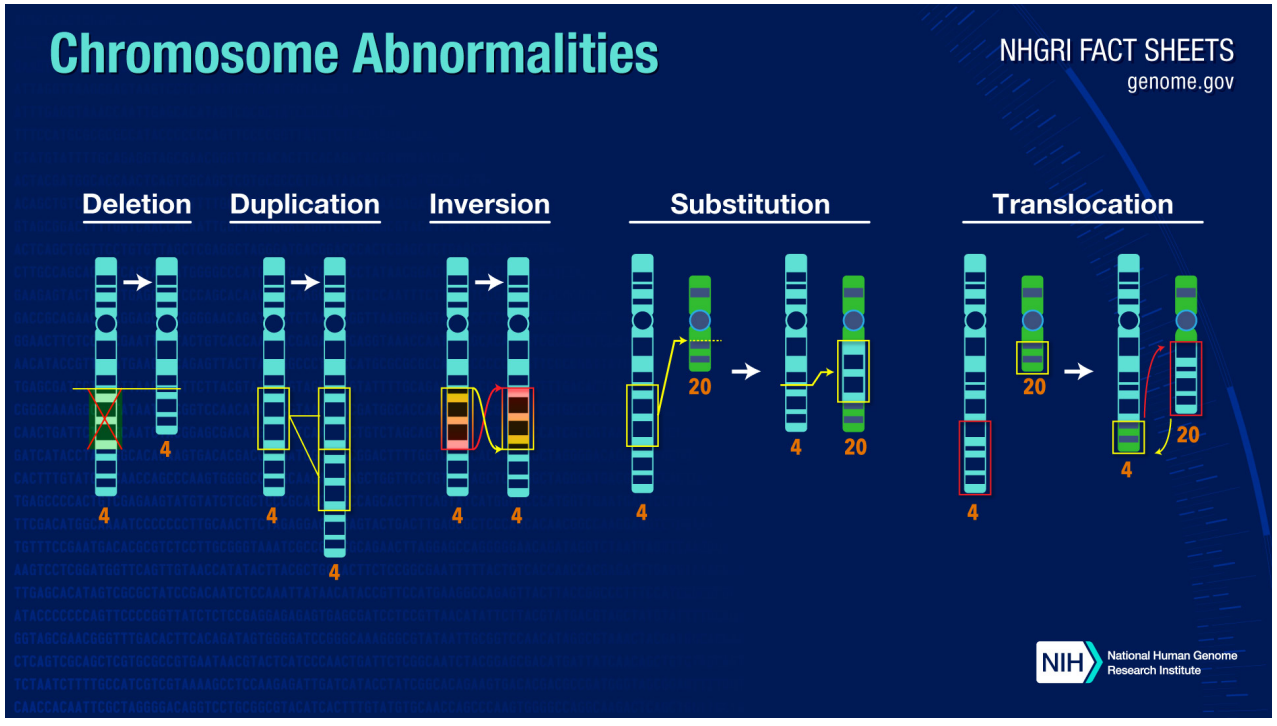
Output: Shortest sequence of reversals turning π into $1, 2, 3 \cdots |\pi|$

Surprisingly, can be solved in polynomial-time [Hannenhalli–Pevzner, JACM 1999]
Distance can even be obtained in linear time! [Baden-Moret-Yan, JCB 2004]
At the origin of multiple works on sorting (structured/signed) permutations.

A simple(?) algorithm: [Hannenhalli–Pevzner 1999] + [Bergeron 2005]

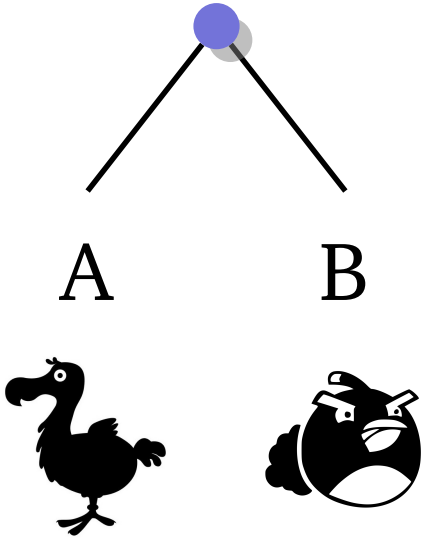
1. Mixed sign (+/-) $\pi \rightarrow$ Greedy reversals inducing max. #oriented pairs
2. Positive sign $\pi \rightarrow$ Elimination (cut/merge $\rightarrow$ reversals) of hurdle(s)

More details to come (lab assignment)

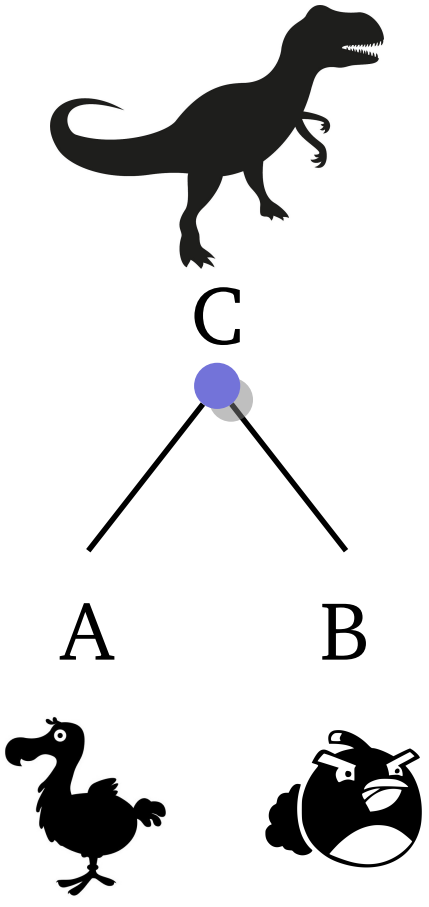


- Fancy operations:
  - Translocations: Exchange ends of chromosomes → Linear time
  - Transpositions: Exchange two consecutive sequences of genes → NP-hard
  - Double Cut and Join...
- Gain/loss of genetic material/syntenic blocks
  → Handle multiple copies of genes
- Extension to unsigned permutations → NP-hard.

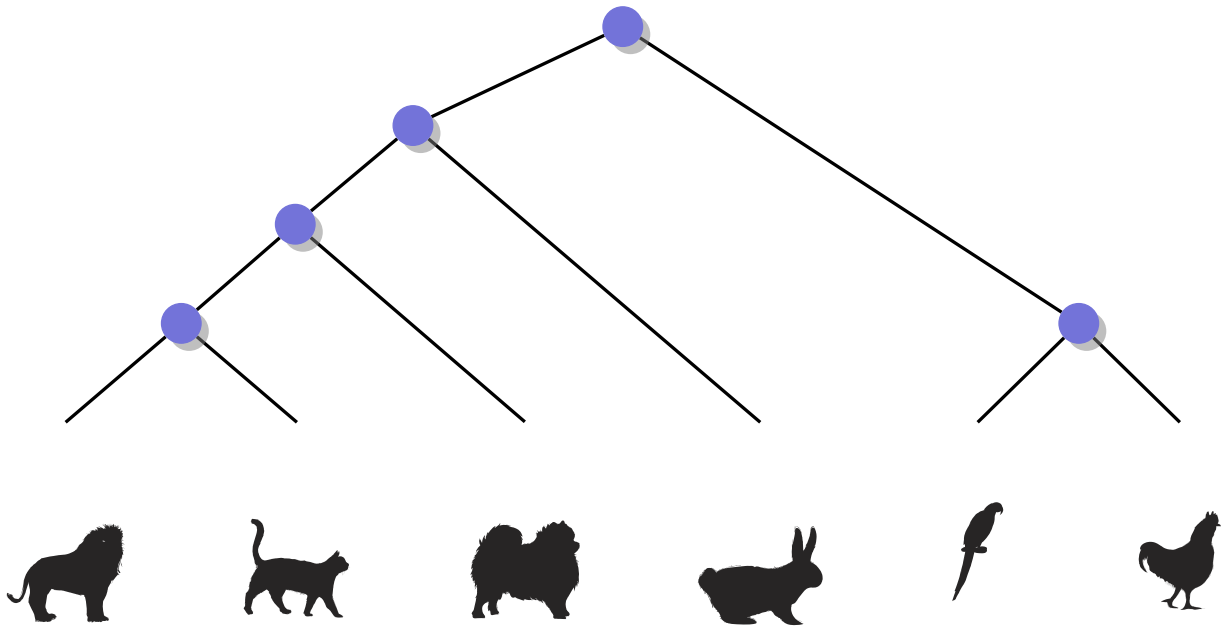
Strong connection to graph theory and (enumerative) combinatorics
→ Very clever/elegant, but **headache** to extrapolate!



Goal: To infer features of ancestral genomes



Goal: To infer features of ancestral genomes

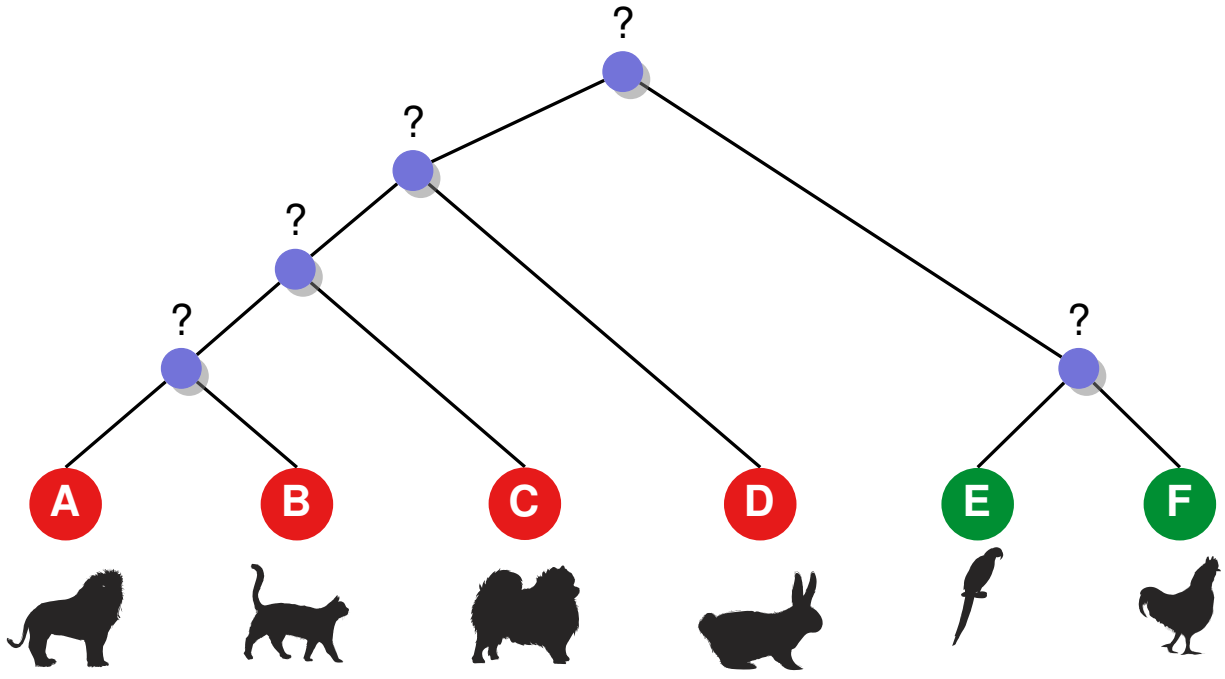


Phylogenetic tree $\mathcal{T} = (V, E)$ + Present/Absent **traits** for extant species
What are the **most likely** features of ancestral species?

Parsimony: Find ancestors labeling $\xi : V \to \{\mathrm{P}, \mathrm{A}\}$ minimizing total change:

$$f(\xi) = \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Example: Min. #event $\to d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

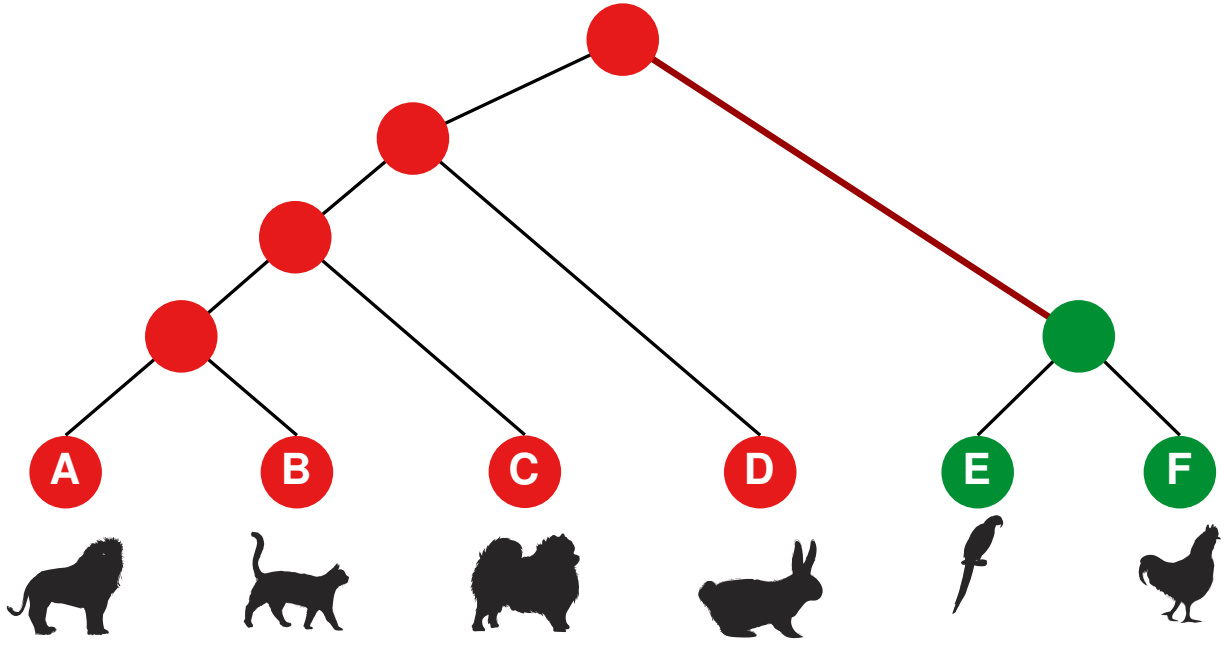


Phylogenetic tree $\mathcal{T} = (V, E)$ + Present/Absent **traits** for extant species
What are the **most likely** features of ancestral species?

Parsimony: Find ancestors labeling $\xi : V \to \{\mathrm{P}, \mathrm{A}\}$ minimizing total change:

$$f(\xi) = \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Example: Min. #event $\to d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

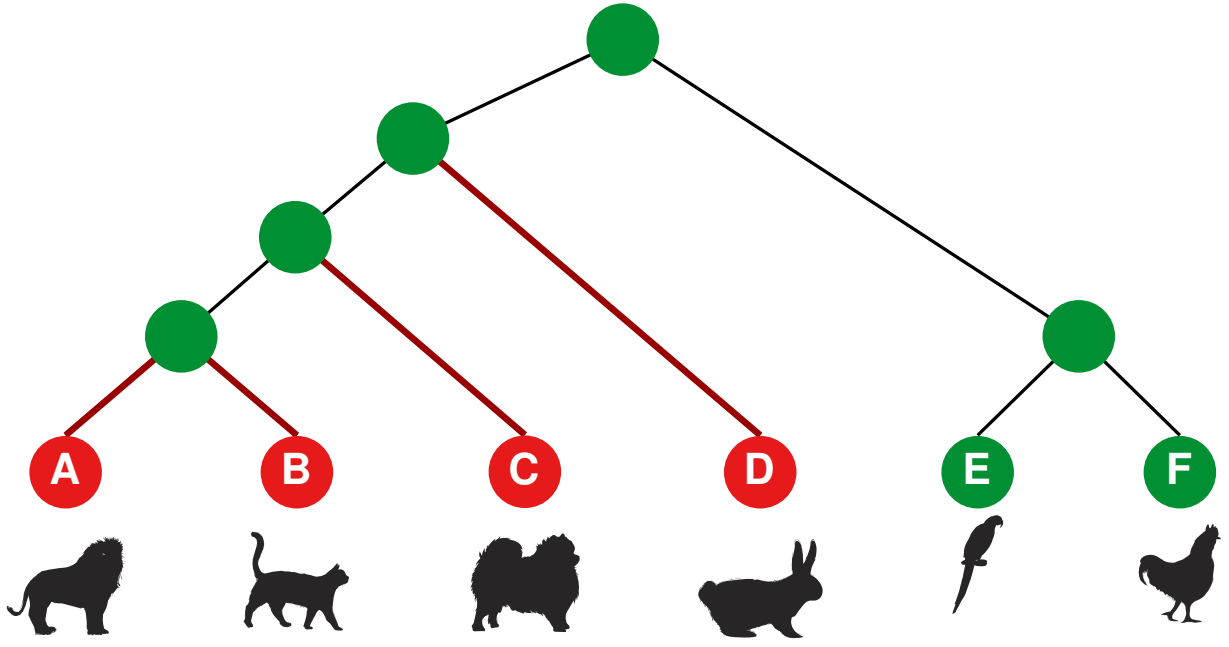


Phylogenetic tree $\mathcal{T} = (V, E)$ + Present/Absent **traits** for extant species
What are the **most likely** features of ancestral species?

Parsimony: Find ancestors labeling $\xi : V \to \{\mathrm{P}, \mathrm{A}\}$ minimizing total change:

$$f(\xi) = \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Example: Min. #event $\to d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

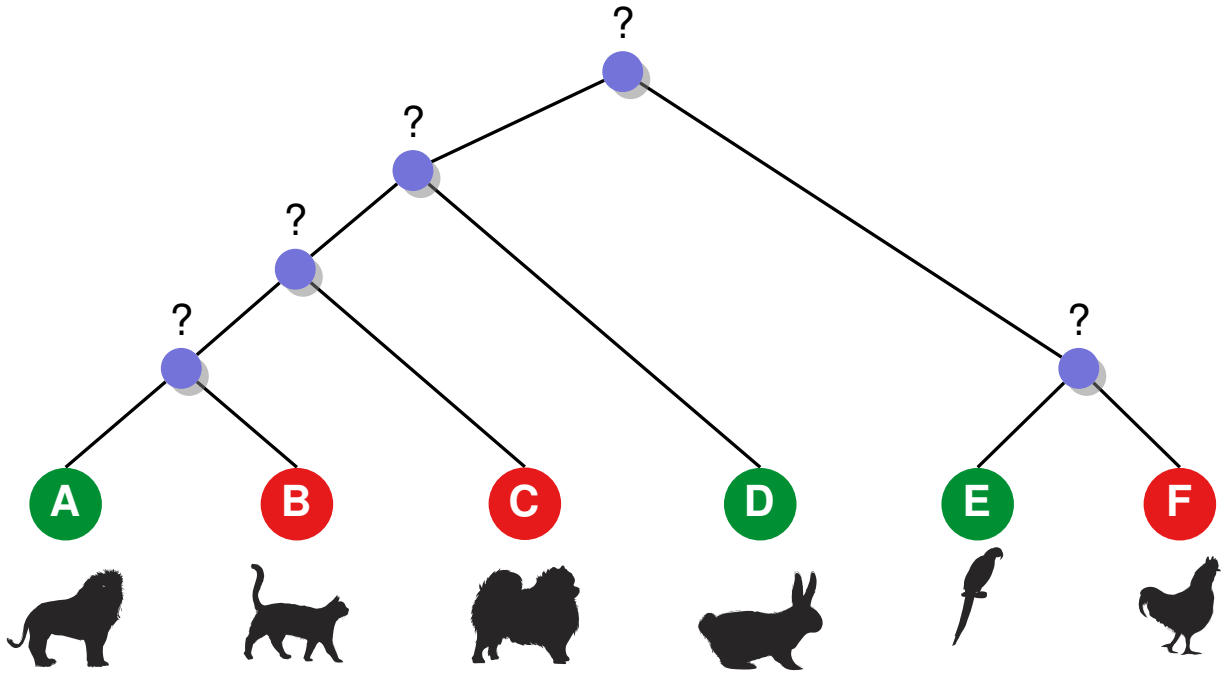


Phylogenetic tree $\mathcal{T} = (V, E)$ + Present/Absent **traits** for extant species
What are the **most likely** features of ancestral species?

Parsimony: Find ancestors labeling $\xi : V \to \{\mathrm{P}, \mathrm{A}\}$ minimizing total change:

$$f(\xi) = \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Example: Min. #event $\to d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise



Phylogenetic tree $\mathcal{T} = (V, E)$ + Present/Absent **traits** for extant species
What are the **most likely** features of ancestral species?

Parsimony: Find ancestors labeling $\xi : V \to \{\mathrm{P}, \mathrm{A}\}$ minimizing total change:

$$f(\xi) = \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Example: Min. #event $\to d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

A labeling ξ is said to **extend** ξ' iff their labels coincide on extant species

PARSIMONY LABELING **problem**

Input: Species tree $\mathcal{T}$; Labeling $\xi' : V \to \{\mathrm{P}, \mathrm{A}\}$ of extant species

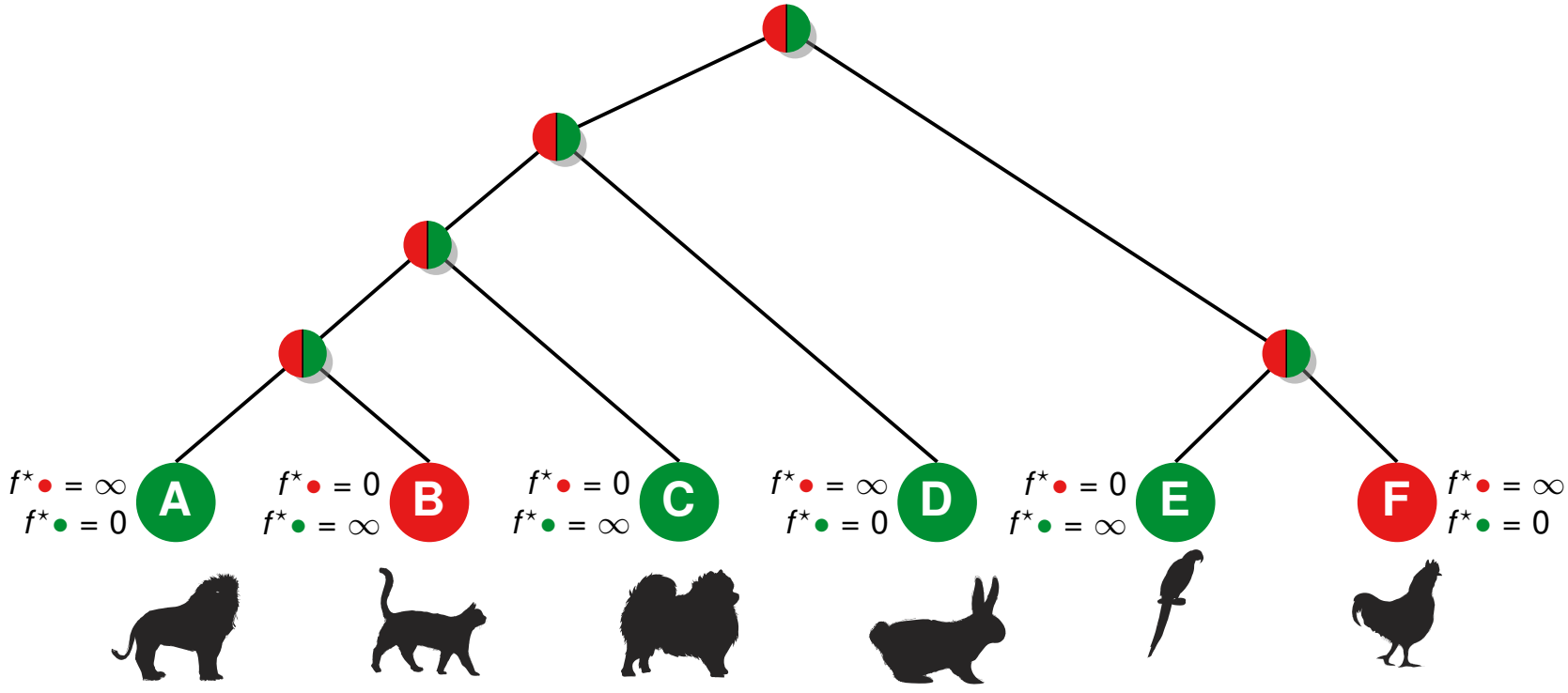
Output: Complete labeling $\xi^\star$ extending ξ' which minimizes total change

$$\xi^\star = \operatorname*{argmin}_{\xi \text{ ext. } \xi'} \sum_{(u,v)\in\mathcal{T}} d(\xi(u), \xi(v))$$

Solved by **dynamic programming** (hey, it's been a while!) [Sankoff/Rousseau 1975]
based on a *hypothetical game*, *played* at any ancestral/internal node u:

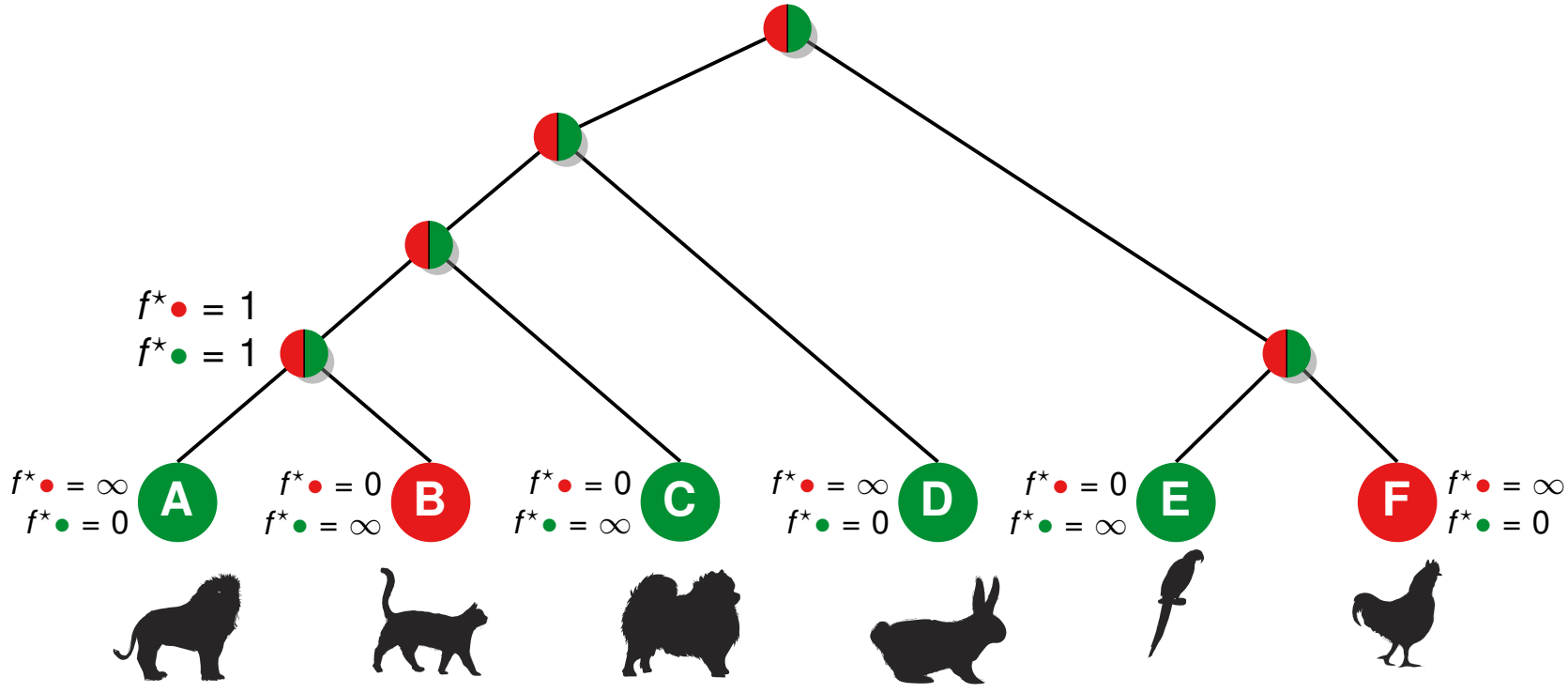
- What if ancestral node u had a given label $x \in \{\mathrm{P}, \mathrm{A}\}$?
- Then min change from u with $\xi(u) = x$ is obtained by minimizing over possible labels y, y' for children v, v' of u, and summing:
  - Direct contribution to change: $d(x, y) + d(x, y')$
  - Min change of a labeling of subtrees v, v', independent of (u, x) (rec.)

$$\to \underset{u \text{ ancestral}}{f_x^\star(u)} = \sum_{uv\in\mathcal{T}} \min_{y\in\{\mathrm{P},\mathrm{A}\}} d(x, y) + f_y^\star(v) \qquad \underset{u \text{ extant/leaf}}{f_x^\star(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



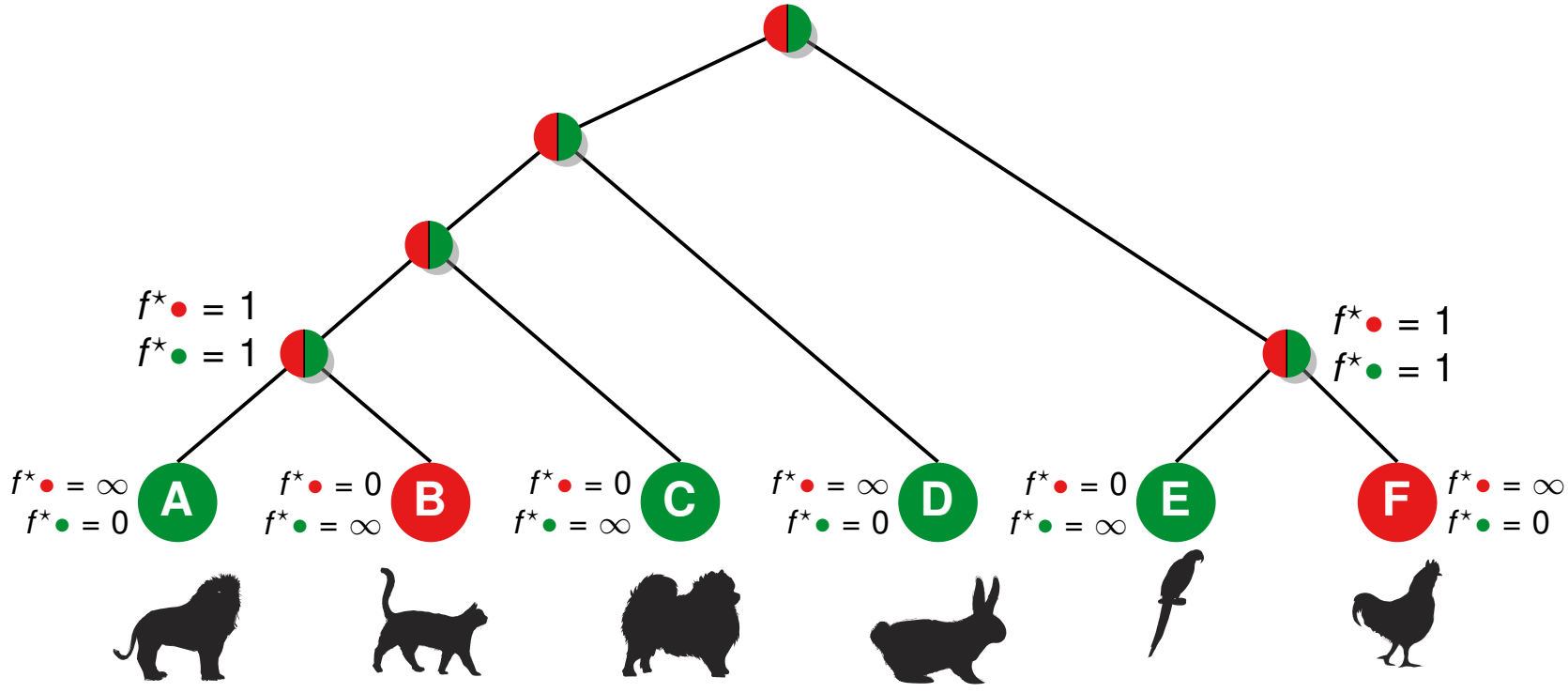
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f_x^\star(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P},\mathrm{A}\}} d(x,y) + f_y^\star(v) \qquad \underset{u \text{ extant/leaf}}{f_x^\star(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



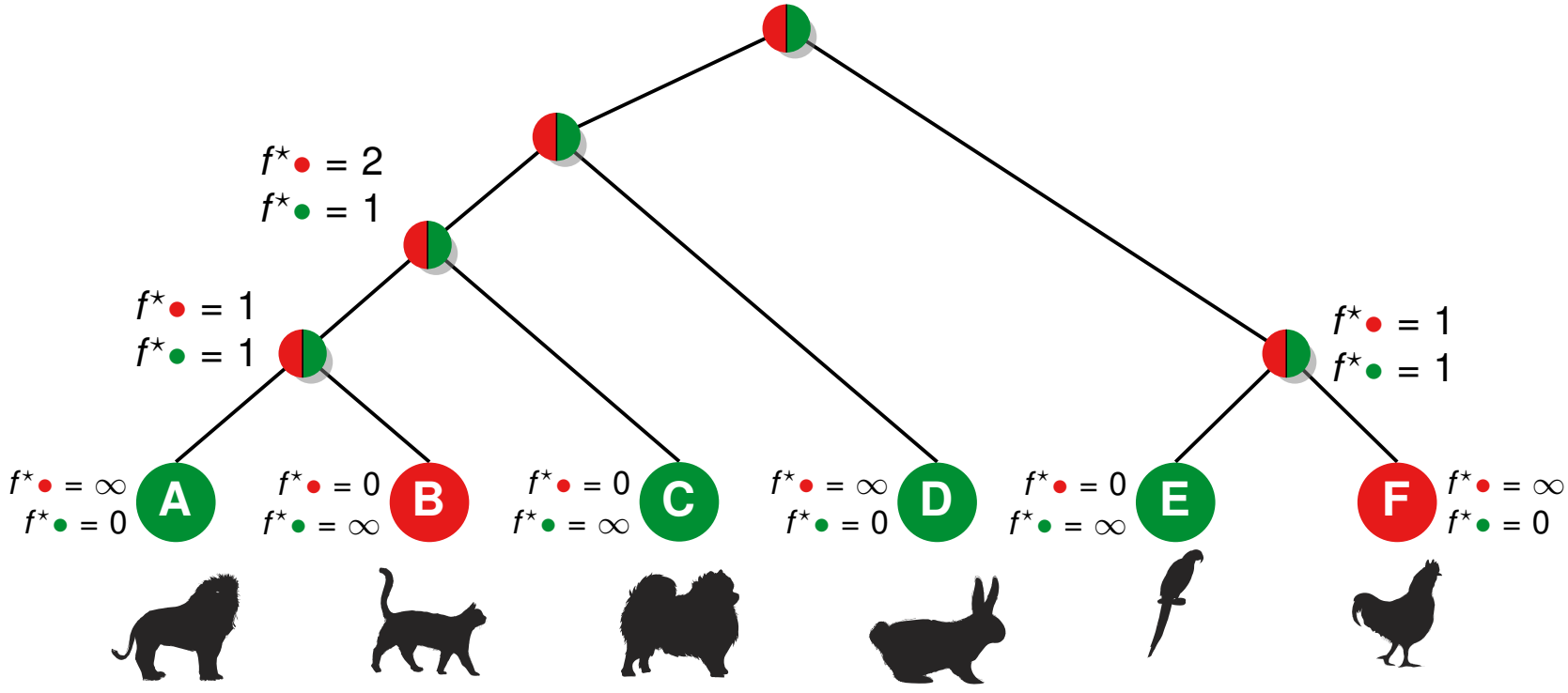
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f_x^\star(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P},\mathrm{A}\}} d(x,y) + f_y^\star(v) \qquad \underset{u \text{ extant/leaf}}{f_x^\star(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



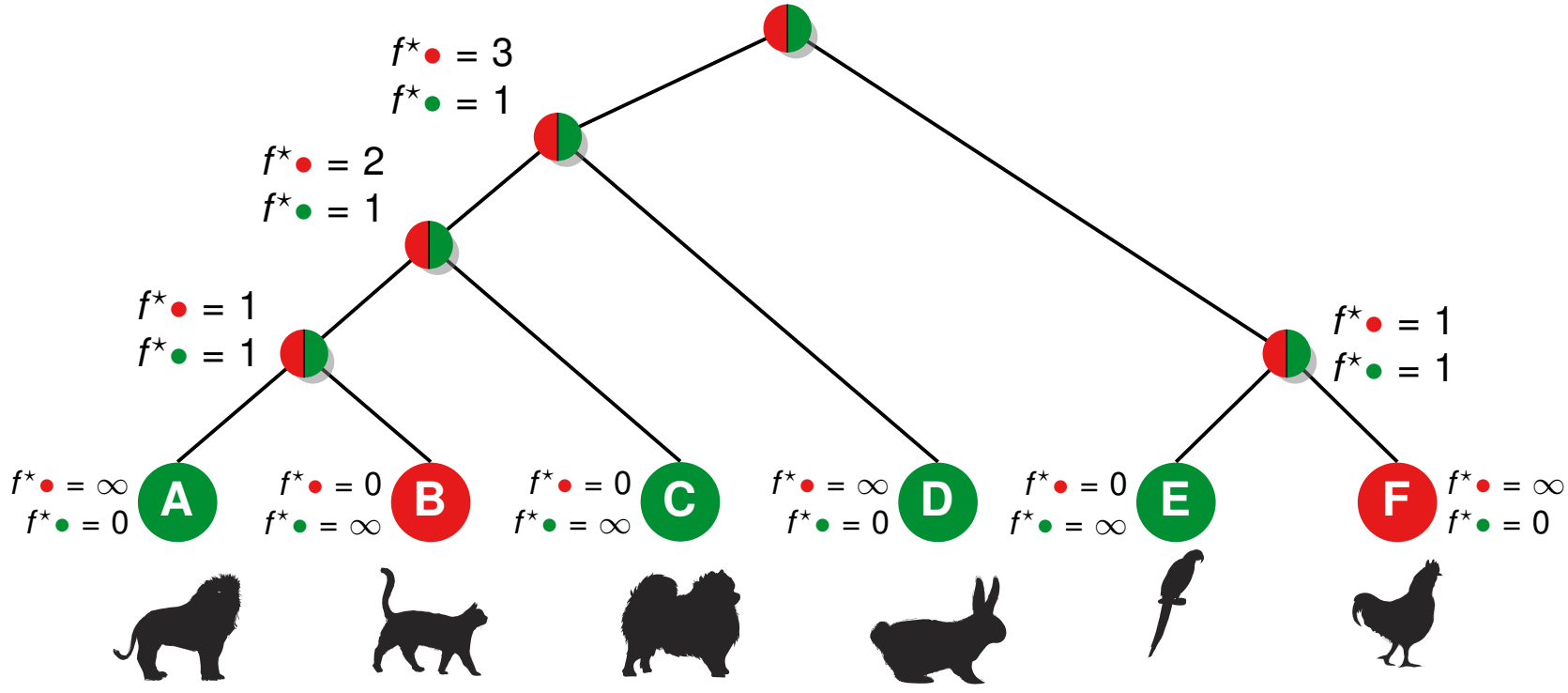
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f_x^\star(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P}, \mathrm{A}\}} d(x, y) + f_y^\star(v) \qquad \underset{u \text{ extant/leaf}}{f_x^\star(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



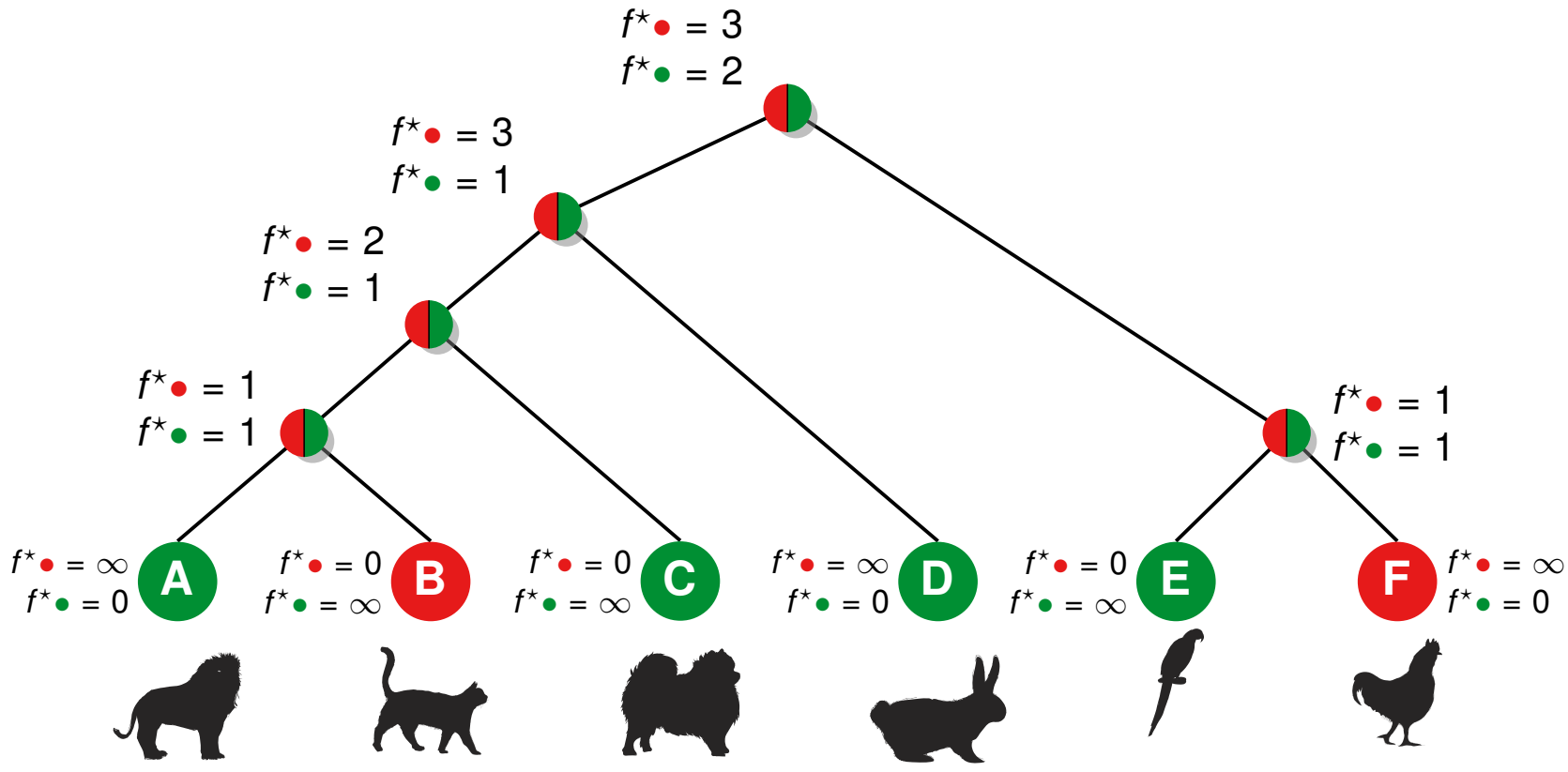
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f^\star_x(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P},\mathrm{A}\}} d(x,y) + f^\star_y(v) \qquad \underset{u \text{ extant/leaf}}{f^\star_x(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



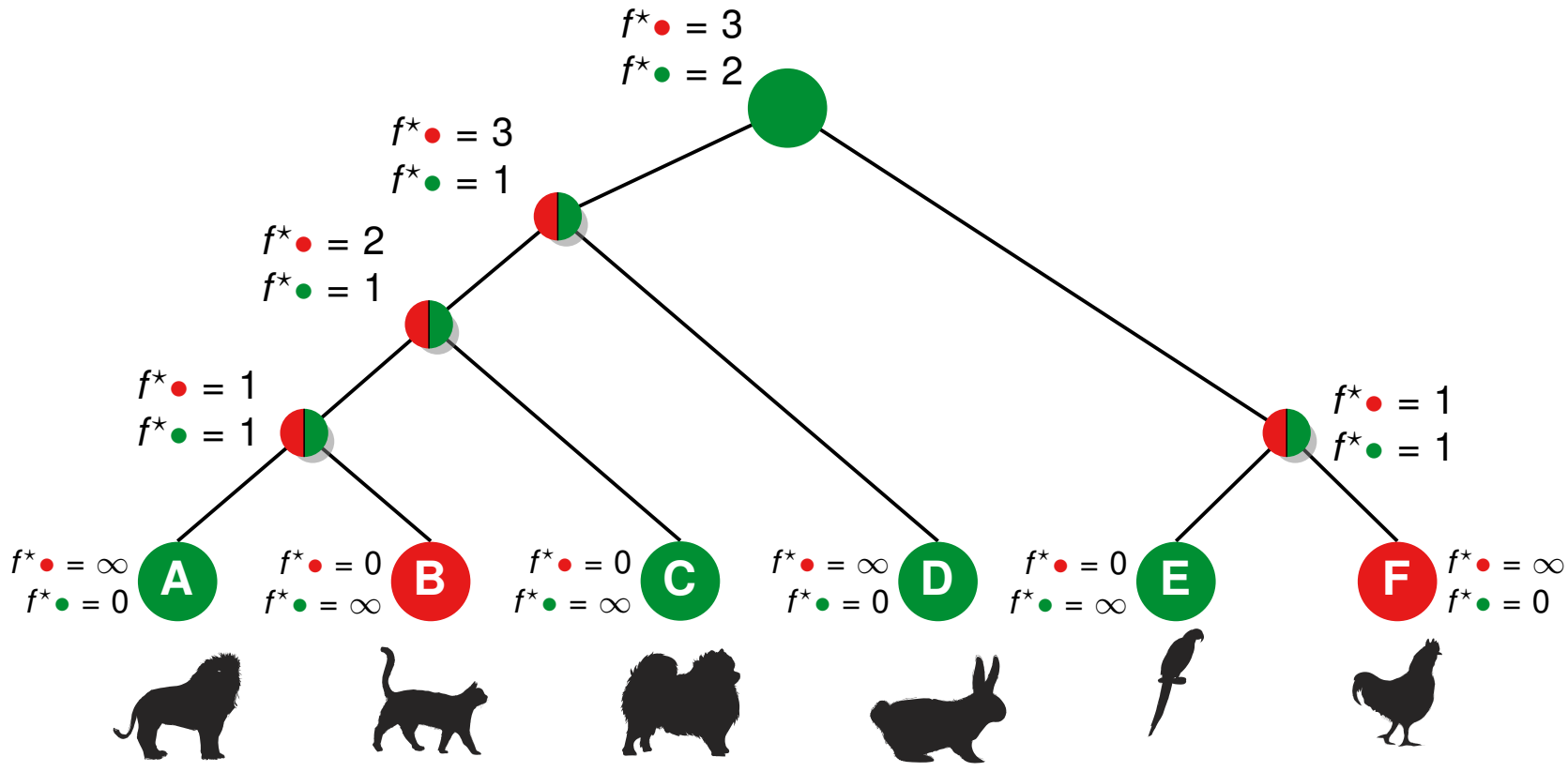
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f^\star_x(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P},\mathrm{A}\}} d(x,y) + f^\star_y(v) \qquad \underset{u \text{ extant/leaf}}{f^\star_x(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



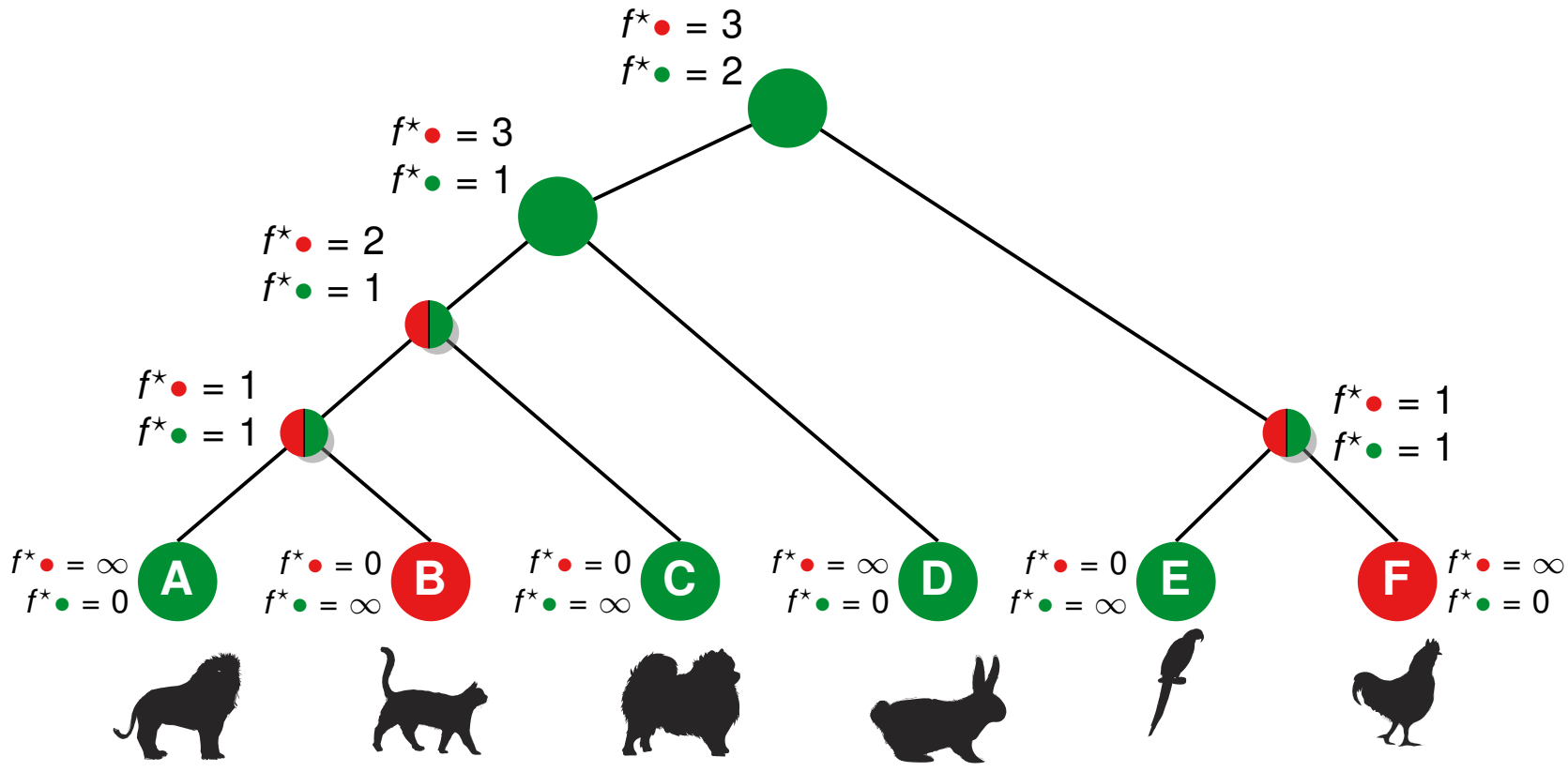
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f^\star_x(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P},\mathrm{A}\}} d(x,y) + f^\star_y(v) \qquad \underset{u \text{ extant/leaf}}{f^\star_x(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



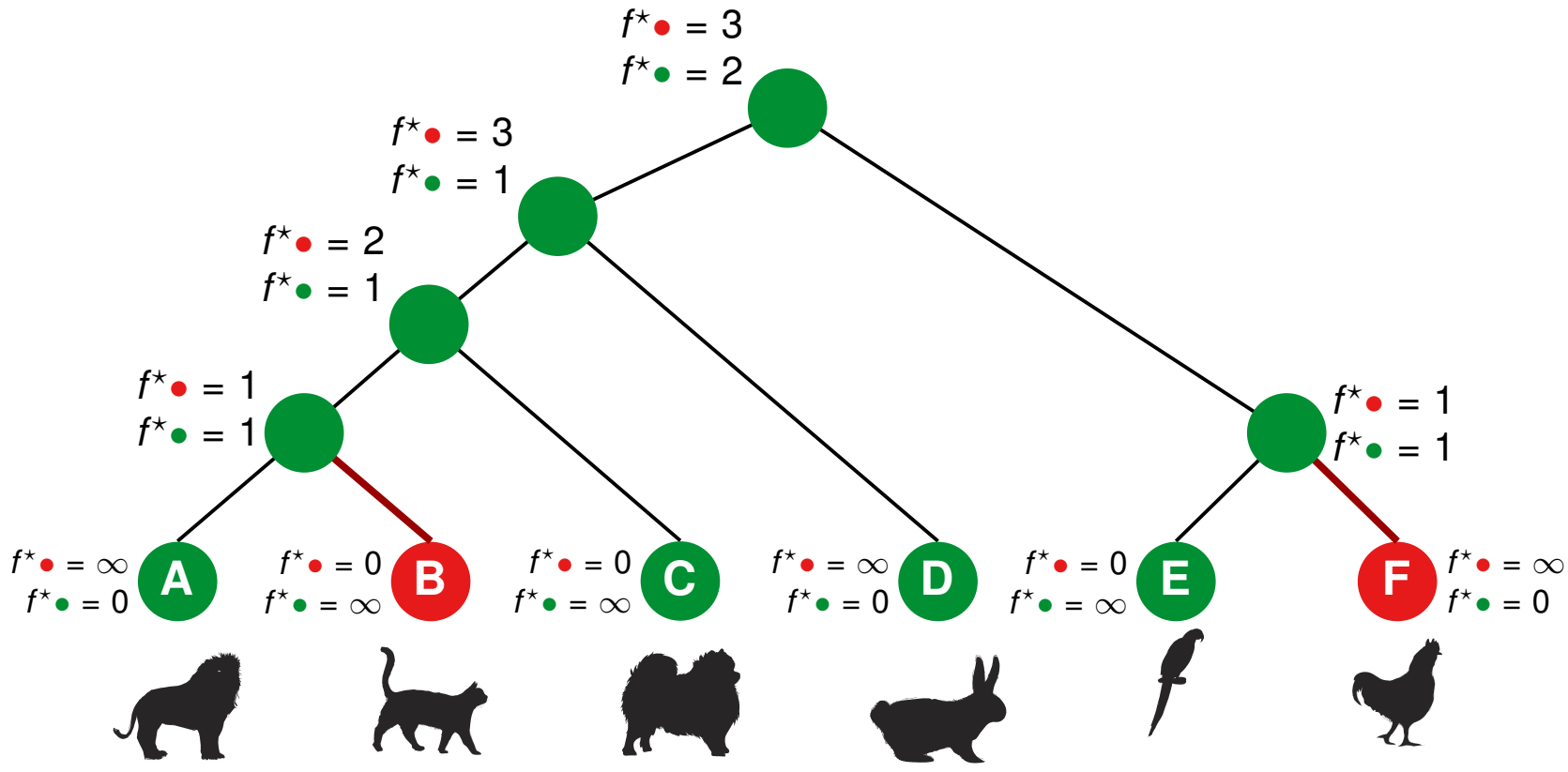
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f^\star_x(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P}, \mathrm{A}\}} d(x, y) + f^\star_y(v) \qquad \underset{u \text{ extant/leaf}}{f^\star_x(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



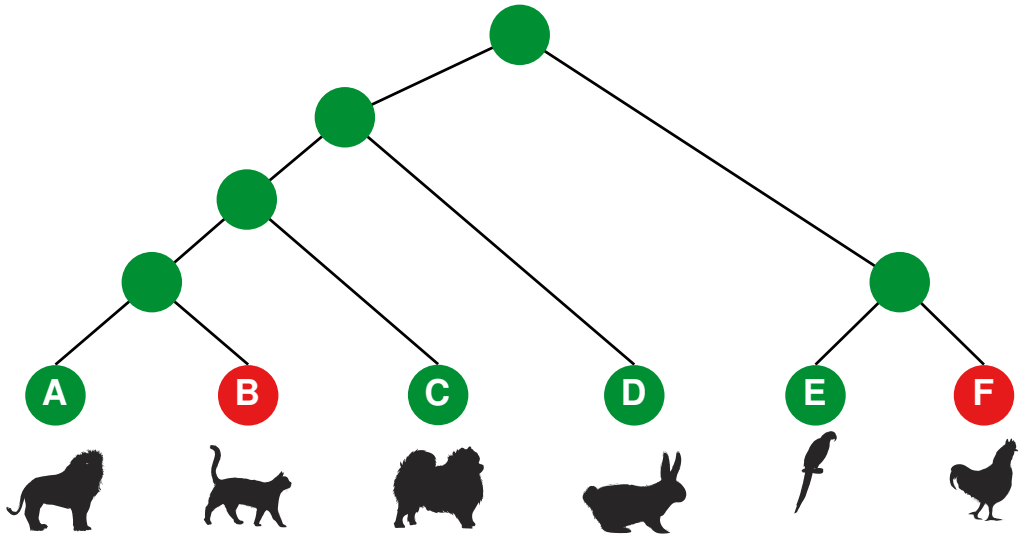
Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

$$\underset{u \text{ ancestral}}{f_x^\star(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\text{P},\text{A}\}} d(x,y) + f_y^\star(v) \qquad \underset{u \text{ extant/leaf}}{f_x^\star(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



Toy model: Min. #event $\rightarrow d(\xi(u), \xi(v)) = 1$ if $\xi(u) \neq \xi(v)$; 0 otherwise

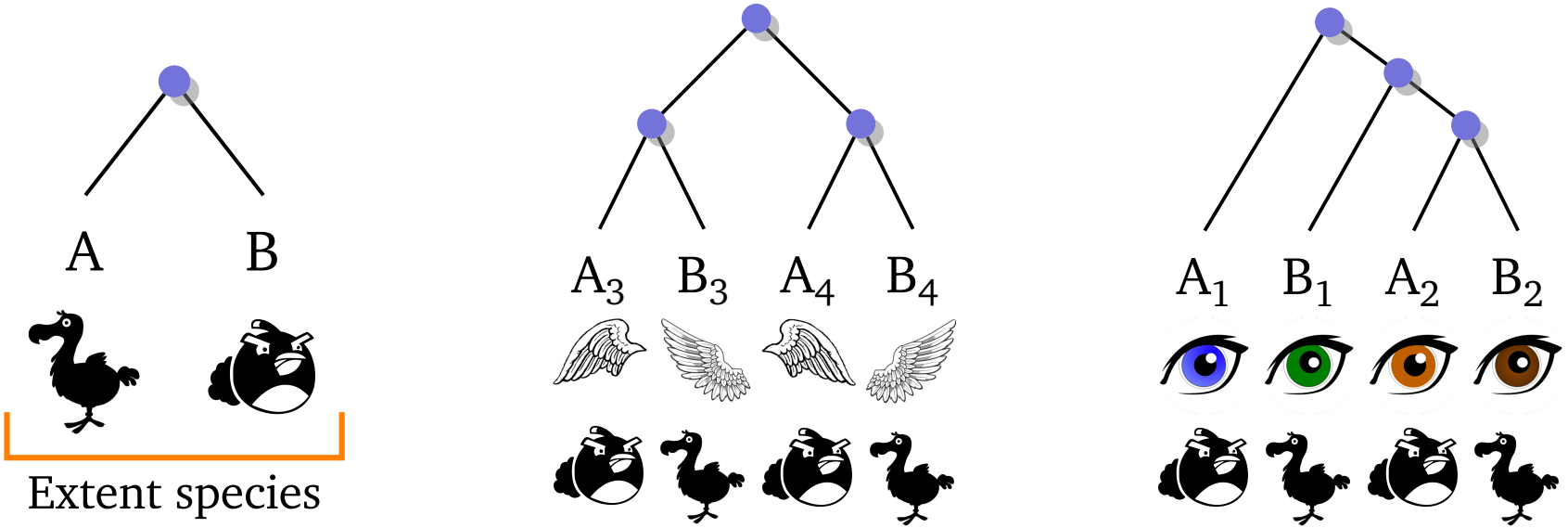
$$\underset{u \text{ ancestral}}{f^\star_x(u)} = \sum_{uv \in \mathcal{T}} \min_{y \in \{\mathrm{P}, \mathrm{A}\}} d(x, y) + f^\star_y(v) \qquad \underset{u \text{ extant/leaf}}{f^\star_x(u)} = \begin{cases} 0 & \text{if } x = \xi'(u) \\ +\infty & \text{otherwise} \end{cases}$$



Sankoff/Rousseau parsimony represents a very general framework, which can be used to infer:

- presence/absence of regulatory motifs
- ancestral sequences
- ancestral structures (Sankoff algo. for RNA comparative folding)
- ancestral genome architectures (adjacencies, gene loss/gain)
- ancestral allele frequencies (∞ #labels $\rightarrow$ restriction on d)

Much can be learned from apparent contradictions. . .

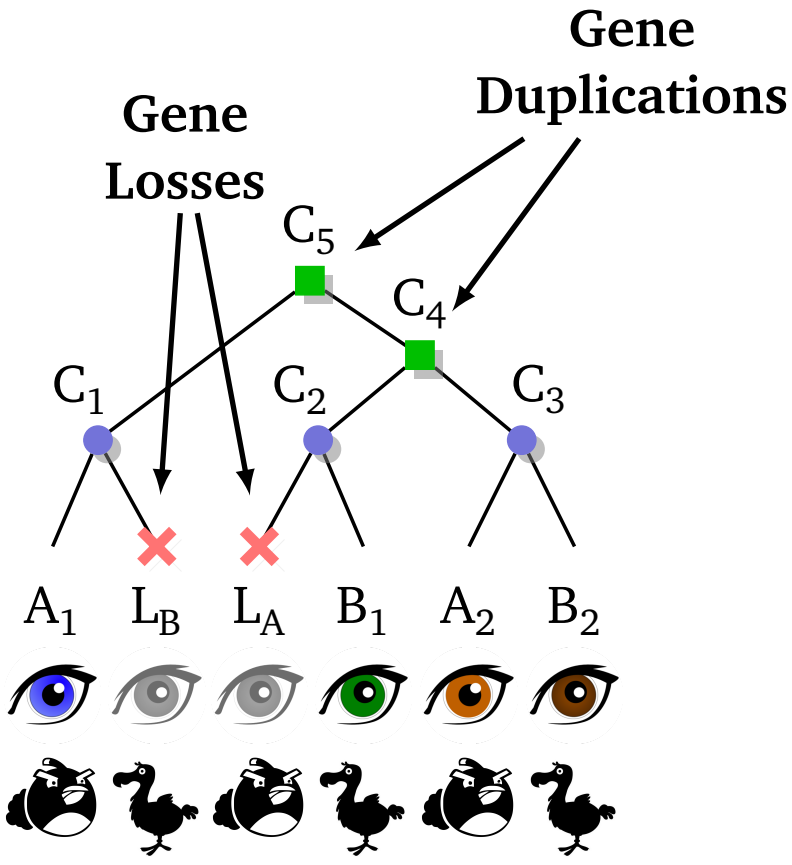


Species tree

Gene trees

How best to **explain** those topological discrepancies?

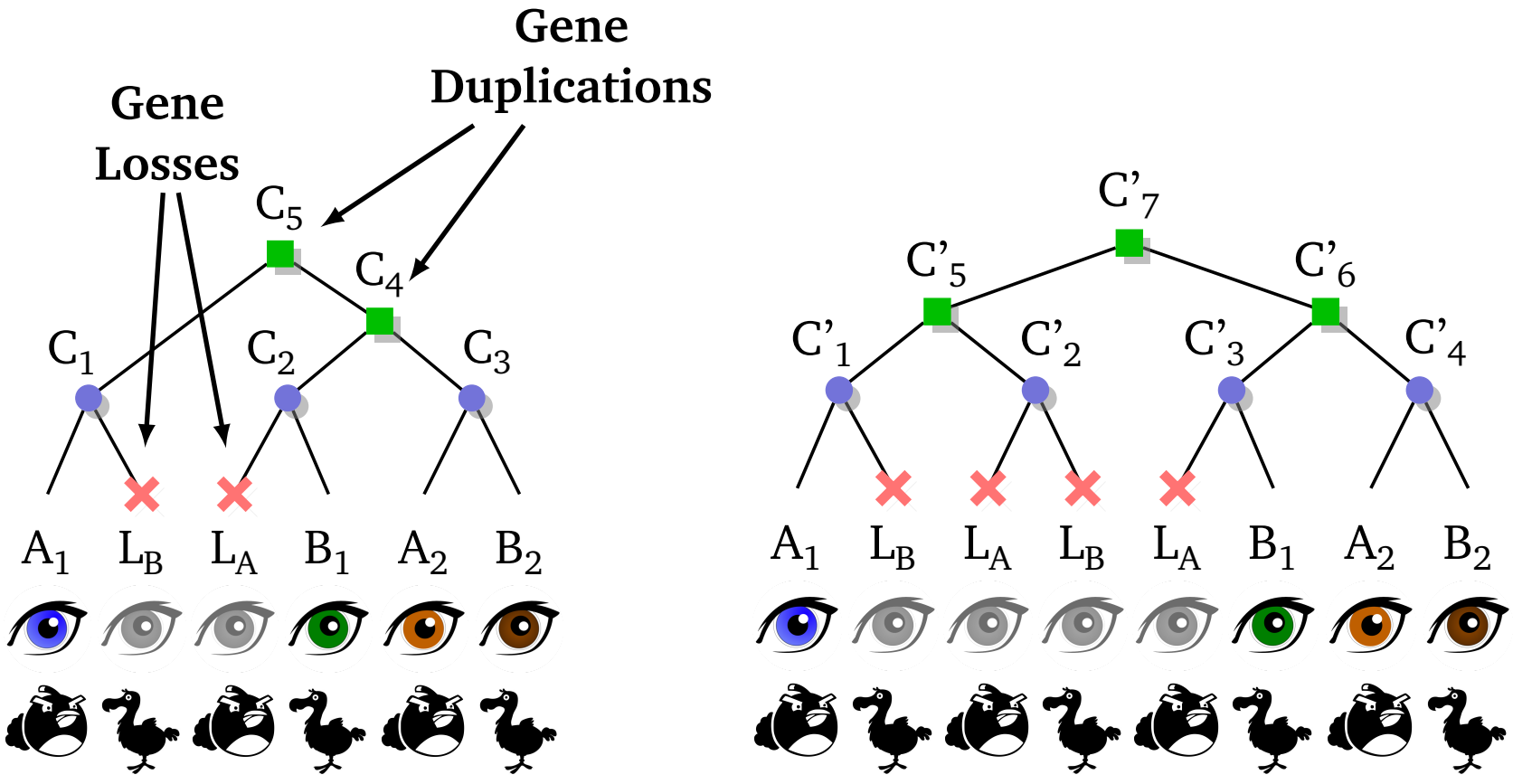
and **learn** something along the way



Ancestral events (duplication/losses/...) induce **apparent** inconsistencies

Reconciliations represent evolutionary scenario that **explain** gene trees by embedding them within species trees **but** some more *reasonable* than others

Goal: Find **parsimonious** reconciliation



Ancestral events (duplication/losses/...) induce **apparent** inconsistencies

Reconciliations represent evolutionary scenario that **explain** gene trees by embedding them within species trees **but** some more *reasonable* than others

Goal: Find **parsimonious** reconciliation

Duplication/Loss reconciliation: Binary tree R such that :

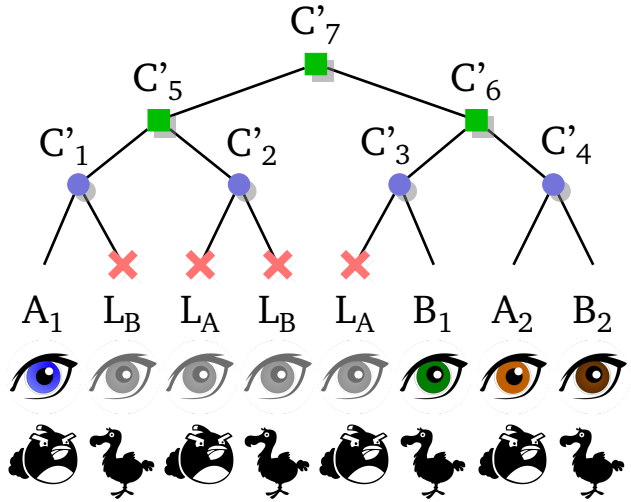
- leaf labeled with extant gene or Loss operation;
- internal node labeled with Dup or Spec event

All nodes are mapped through σ to a species

R **congruent** with **gene tree** G if removing Loss from R + contract 1^{ary} branches $\rightarrow G$

R **congruent** with **species tree** S if:

- For each Spec node $u \rightarrow v, w$ in R, we have $\sigma_u \rightarrow \sigma_v, \sigma_w$ in S
- For each Dup node $u \rightarrow v, w$ in R, we have $\sigma_u = \sigma_v = \sigma_w$



DL RECONCILIATION **problem**

Input: Gene tree G, Species tree S

Output: Reconciliation R **congruent** to both G and S, minimizing cost

$$\text{Cost}(R) = \alpha \times \#\text{Loss}(R) + \beta \times \#\text{Dup}(R)$$

Can be solved efficiently using LCA mapping... [Page, Syst. Biol. 1994]

...or using dynamic programming in $\mathcal{O}(|S| \times |G|)$ time/space

DL RECONCILIATION **problem**

Input: Gene tree G, Species tree S

Output: Reconciliation R, **congruent** to G and S, minimizing cost

$$\text{Cost}(R) = \alpha \times \#\text{Loss}(R) + \beta \times \#\text{Dup}(R)$$

Notations:

$C[g, s]$ Best cost of reconciliation for node/subtrees $g \sqsubseteq G$ and $s \sqsubseteq S$

L_u, R_u Left/right child of node u

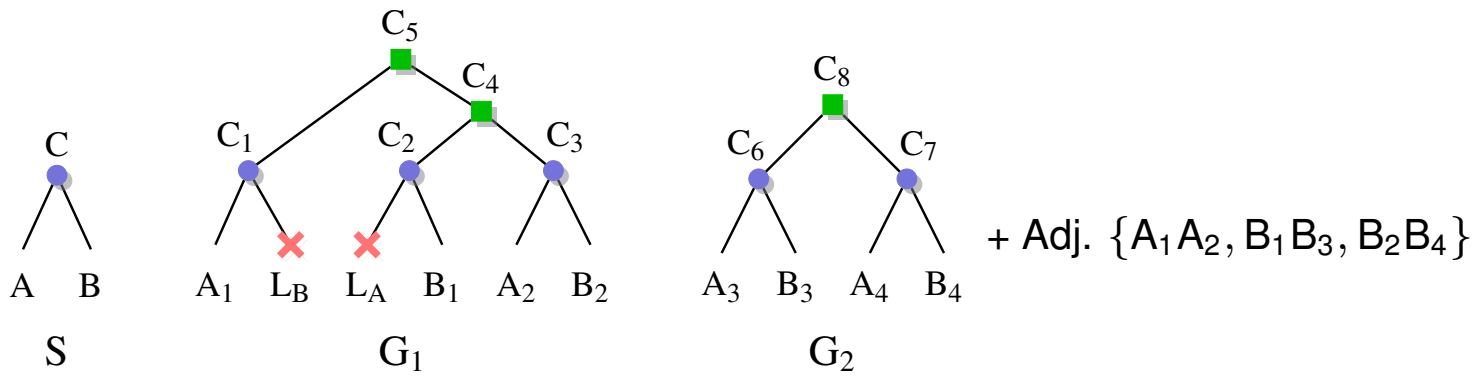
$\sigma(u)$ Species of a leaf u within gene tree (or species tree... duh!)

$$\underset{g,s \text{ leaves}}{C[g, s]} = \begin{cases} 0 & \text{if } \sigma(g) = \sigma(s) \\ +\infty & \text{otherwise} \end{cases} \qquad \underset{g \text{ leaf}, s \text{ internal}}{C[g, s]} = +\infty$$

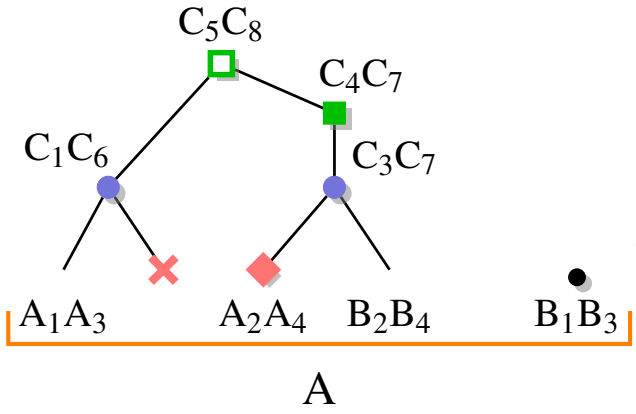
$$C[g, s] = \begin{cases} \beta + C[L_g, s] + C[R_g, s] & \text{Ancestor gene } g \text{ duplicated in species } s \\ C[L_g, L_s] + C[R_g, R_s] & \text{Speciation event, direct mapping} \\ C[L_g, R_s] + C[R_g, L_s] & \text{Speciation event, crossed mapping} \\ \alpha + C[g, L_s] & \text{Loss of gene subtree } g \text{ in right part of species tree} \\ \alpha + C[g, R_s] & \text{Loss of gene subtree } g \text{ in left part of species tree} \end{cases}$$

- Efficient algorithms exist for parsimonious scenarios for genomic characters (genes under DTL, for instance).
- Bérard *et al* (2012) extended methods to gene adjacencies under gene Speciation, Duplication, Loss.
- DeCo considers cost of gains/ breaks due to rearrangements.

Problem: Given two reconciliations (R_1, R_2) + extant adjacencies, construct parsimonious evolutionary scenario



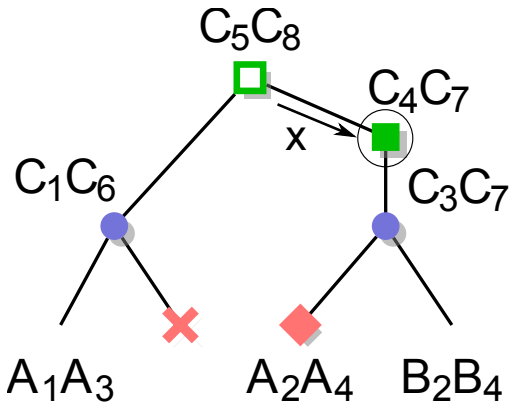
Output: Adjacency forest



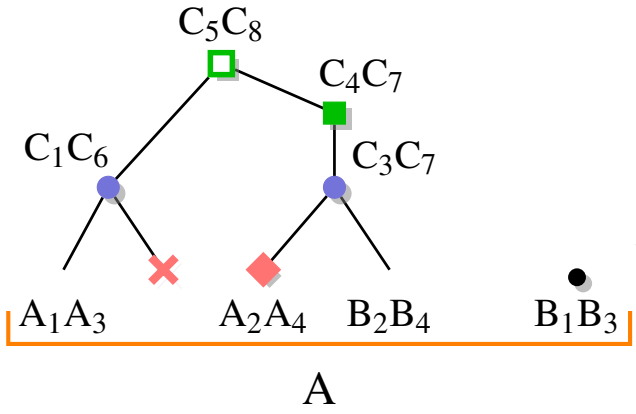
→ Ancestral genome architecture

Event costs/parameters

- Adjacency gain: x
- Adjacency break: y



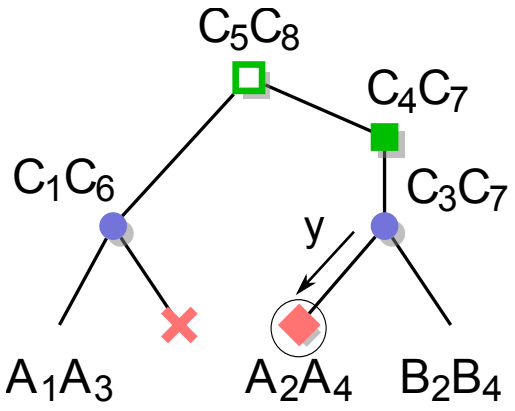
Output: Adjacency forest



→ Ancestral genome architecture

Event costs/parameters

- Adjacency gain: x
- Adjacency break: y



1. $e(g_1) = \text{Extant}$ and $e(g_2) = \text{Extant}$:

$$c_1(g_1, g_2) = \begin{cases} 0 & \text{if } g_1 g_2 \text{ is an adjacency} \\ 0 & \text{otherwise} \end{cases}$$

$$c_0(g_1, g_2) = \begin{cases} 0 & \text{if } g_1 g_2 \text{ is not an adjacency} \\ 0 & \text{otherwise} \end{cases}$$

2. $e(g_1) = \text{GLoss}$ and $e(g_2) \in \{\text{Extant}, \text{Spec}, \text{GDup}\}$:

$$c_1(g_1, g_2) = c_0(g_1, g_2) = 0^{\#\text{NonGDup}(g_2)}$$

3. $e(g_1) \in \{\text{Extant}, \text{Spec}, \text{GDup}\}$ and $e(g_2) = \text{GLoss}$:

$$c_1(g_1, g_2) = c_0(g_1, g_2) = 0^{\#\text{NonGDup}(g_1)}$$

4. $e(g_1) = \text{GLoss}$ and $e(g_2) = \text{GLoss}$:

$$c_1(g_1, g_2) = c_0(g_1, g_2) = 0$$

5. $e(g_1) \in \{\text{Extant}, \text{Spec}\}$ and $e(g_2) = \text{GDup}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(g_1, b_{g_2}) + c_0(g_1, a_{g_2}), & c_0(g_1, b_{g_2}) + c_1(g_1, a_{g_2}), \\ c_1(g_1, b_{g_2}) + c_1(g_1, a_{g_2}) + x, & c_0(g_1, b_{g_2}) + c_0(g_1, a_{g_2}) + y \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(g_1, b_{g_2}) + c_0(g_1, a_{g_2}), & c_0(g_1, b_{g_2}) + c_1(g_1, a_{g_2}) + x, \\ c_1(g_1, b_{g_2}) + c_0(g_1, a_{g_2}) + x, & c_1(g_1, b_{g_2}) + c_1(g_1, a_{g_2}) + 2x \end{cases}$$

6. $e(g_1) = \text{GDup}$ and $e(g_2) \in \{\text{Extant}, \text{Spec}\}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a_{g_1}, g_2) + c_0(b_{g_1}, g_2), & c_0(a_{g_1}, g_2) + c_1(b_{g_1}, g_2), \\ c_1(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + x, & c_0(a_{g_1}, g_2) + c_0(b_{g_1}, g_2) + y \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a_{g_1}, g_2) + c_0(b_{g_1}, g_2), & c_0(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + x, \\ c_1(a_{g_1}, g_2) + c_0(b_{g_1}, g_2) + x, & c_1(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + 2x, \end{cases}$$

7. $e(g_1) = \text{Spec}$ and $e(g_2) = \text{Spec}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + 0, & c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + y + 0, \\ c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + y + 0, & c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + 2y + 0, \\ c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + 0, & c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + y + 0, \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + y + 0, & c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + 2y + 0 \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + 0, & c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + x + 0, \\ c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x + 0, & c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + 2x + 0, \\ c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + 0, & c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + x + 0, \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + x + 0, & c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + 2x + 0, \end{cases}$$

8. $e(g_1) = \text{GDup}$ and $e(g_2) = \text{GDup}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a_{g_1}, g_2) + c_0(b_{g_1}, g_2), & c_0(a_{g_1}, g_2) + c_1(b_{g_1}, g_2), \\ c_1(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + x, & c_0(a_{g_1}, g_2) + c_0(b_{g_1}, g_2) + y, \\ c_1(g_1, a_{g_2}) + c_0(g_1, b_{g_2}), & c_0(g_1, a_{g_2}) + c_1(g_1, b_{g_2}), \\ c_1(g_1, a_{g_2}) + c_1(g_1, b_{g_2}) + x, & c_0(g_1, a_{g_2}) + c_0(g_1, b_{g_2}) + y, \\ c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}), & \\ c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x, & \\ c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + x, & \\ c_1(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + 2x, & \\ c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + y, & \\ c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x + y, & \\ c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + x + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + x + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}), & \\ c_0(a_{g_1}, a_{g_2}) + c_1(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x, & \\ c_1(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + x, & \\ c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_1(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_1(b_{g_1}, a_{g_2}) + y, & \\ c_0(a_{g_1}, a_{g_2}) + c_0(b_{g_1}, b_{g_2}) + c_0(a_{g_1}, b_{g_2}) + c_0(b_{g_1}, a_{g_2}) + 2y, & \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a_{g_1}, g_2) + c_0(b_{g_1}, g_2), & c_0(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + x, \\ c_1(a_{g_1}, g_2) + c_0(b_{g_1}, g_2) + x, & c_1(a_{g_1}, g_2) + c_1(b_{g_1}, g_2) + 2x, \\ c_0(g_1, a_{g_2}) + c_0(g_1, b_{g_2}), & c_0(g_1, a_{g_2}) + c_1(g_1, b_{g_2}) + x, \\ c_1(g_1, a_{g_2}) + c_0(g_1, b_{g_2}) + x, & c_1(g_1, a_{g_2}) + c_1(g_1, b_{g_2}) + 2x. \end{cases}$$

Deco algorithm [Berard *et al*, 2012]

Dyn. prog. scheme for most parsimonious adjacency forest in $\mathcal{O}(|R_1||R_2|)$.

.
.
.

7. If $e(g_1) = \text{Spec}$ and $e(g_2) = \text{Spec}$:

$$c_1(g_1, g_2) = \min \begin{cases} c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)),\ c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \text{ABreak}, \\ c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \text{ABreak}, \\ c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \text{2ABreak}, \\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)),\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + \text{ABreak}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + \text{ABreak},\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + \text{2ABreak} \end{cases}$$

$$c_0(g_1, g_2) = \min \begin{cases} c_0(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)),\ c_1(a(g_1), b(g_2)) + c_0(b(g_1), a(g_2)) + \text{AGain}, \\ c_0(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \text{AGain},\ c_1(a(g_1), b(g_2)) + c_1(b(g_1), a(g_2)) + \text{2AGain}, \\ c_0(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)),\ c_1(a(g_1), a(g_2)) + c_0(b(g_1), b(g_2)) + \text{AGain}, \\ c_0(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + \text{AGain},\ c_1(a(g_1), a(g_2)) + c_1(b(g_1), b(g_2)) + \text{2AGain} \end{cases}$$

8. If $e(g_1) = \text{GDup}$ and $e(g_2) = \text{GDup}$:

.
.
.

Deco algorithm [Berard *et al*, 2012]

Dyn. prog. scheme for most parsimonious adjacency forest in $\mathcal{O}(|R_1| \times |R_2|)$

Intuition: Jointly explores both reconciliations in all possible ways, to account for adjacency breaks/gains.

Comparative genomics is a vibrant field with strong **2-ways** links with

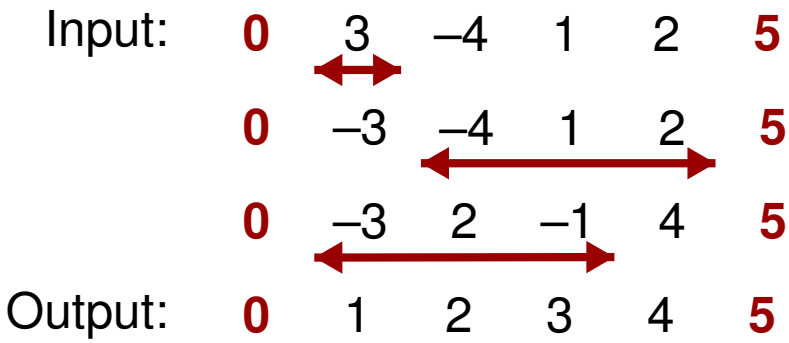
- Phylogenetics, (Epi)genomics, Paleogenomics, Proteomics, Structural biology, Systems Biology . . .
- . . . but also string processing, enumerative combinatorics, discrete algorithms , discrete maths . . .

It predicts the past, but also informs studies of the unobservable present

Some $\pm$ current directions:

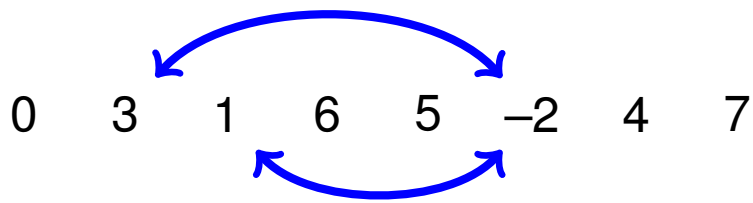
- Beyond parsimony, Bayesian approaches (likelihood maximization) $\rightarrow$ Mainly based on sampling (ABC), but also more direct opt. (DP)
- Distinguishing between erroneous and informative incongruences
- Expressive models for reconciliations: Horizontal gene transfers, polytomies. . .
- Integrating multiple steps of analyses into joint optimizations *e.g.* comparative genome scaffolding
- Advanced algorithms: FPT algo/kernels, guaranteed approximations, exp. time (small exponents), ILP. . . . . for all of the above (+ML)

Goal: Implement **sorting by reversal** for signed permutations [Bergeron 2006]
w.l.o.g. assume that input permutation π **flanked** by **0** and **n+1**



Some **definitions**:

- **Oriented pair:** Two pos. (i, j) with consecutive values of opposite signs



Reversal of oriented pairs (i, j):

$\pi_i + \pi_j = +1 \rightarrow$

$$\cdots \quad \pi_i \quad \pi_{i+1} \quad \cdots \quad \pi_{j-1} \quad \pi_j \quad \cdots$$
$$\cdots \quad -\pi_{j-1} \quad -\pi_{j-2} \quad \cdots \quad -\pi_i \quad \pi_j \quad \cdots$$

$\pi_i + \pi_j = -1 \rightarrow$

$$\cdots \quad \pi_i \quad \pi_{i+1} \quad \cdots \quad \pi_{j-1} \quad \pi_j \quad \cdots$$
$$\cdots \quad \pi_i \quad -\pi_j \quad \cdots \quad -\pi_{i+2} \quad -\pi_{i+1} \quad \cdots$$

Goal: Implement **sorting by reversal** for signed permutations [Bergeron 2006]
w.l.o.g. assume that permutations **flanked** by **0** and **n+1**

Definitions for **positive** permutations:

- **Framed interval:** Interval $[i, j]$ in π of the form

  $$\pi_i\ \pi_{i+1}\ \pi_{i+2} \cdots \pi_j \text{ with } \pi_j = \pi_i + (j - i) \text{ and } \pi_i < \pi_k < \pi_j, \forall i < k < j$$

  or **equivalently** region $[i, j]$ contains a permutation of $[\pi_i, \pi_j]$

  But... one needs to consider **circular** ordering of values

Framed intervals: $[1, 8]$, $[2, 6]$, $[6, \bar{2}]$

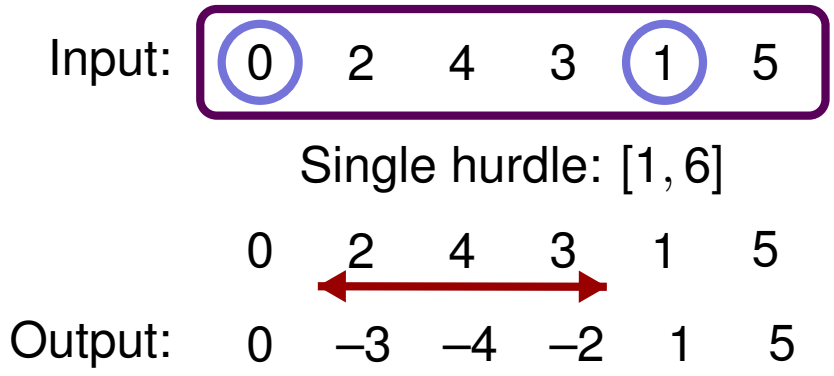
- **Hurdles**: Minimal framed intervals w.r.t. inclusion

Hurdles: $[2, 6]$, $[6, \bar{2}]$

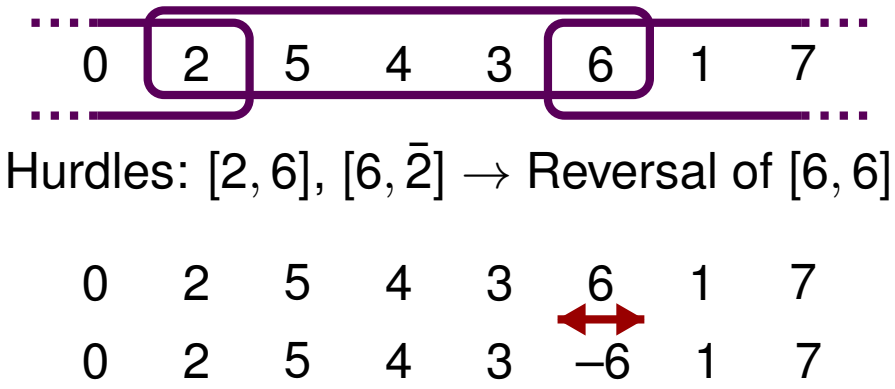
Goal: Implement **sorting by reversal** for signed permutations [Bergeron 2006]
w.l.o.g. assume that permutations **flanked** by **0** and **n+1**

Definitions for **positive** permutations:

- **Splitting a hurdle** $[i, j]$**:** Look for position k such that $\pi_k = \pi_i + 1$, and reverse interval $[i + 1, k - 1]$



- **Merging two hurdles** $[i, j]$ **and** $[k, l]$**,** $j \leq k$**:** Reverse interval $[j, k]$



Goal: Implement **sorting by reversal** for signed permutations [Bergeron 2006]

Final algo. While π not sorted:

- If π negative:
  - Identify Oriented Pair (OP) (i, j) that creates max #OPs
  - Reverse OP (i, j)
  - Iterate until permutation π becomes positive
- If π positive, consider hurdles $\mathcal{H}$ of π:
  - $\mathcal{H} = \{(i, j)\}$
    $\rightarrow$ split (i, j)
  - $\mathcal{H} = \{(i, j), (k, l)\}$
    $\rightarrow$ merge (i, j)
  - $|\mathcal{H}| > 2$, $|\mathcal{H}|$ even
    $\rightarrow$ Merge two non-consecutive hurdles
  - $|\mathcal{H}| > 2$, $|\mathcal{H}|$ odd
    Simple hurdle $h^\star$: Cutting $h^\star$ + reversal of OPs **decreases** #hurdles

    If $\mathcal{H}$ contains simple hurdle $h^\star$
    $\rightarrow$ Cut $h^\star$
    Otherwise
    $\rightarrow$ Merge two non-consecutive hurdles (or consecutive ones if $|\mathcal{H}| = 3$)