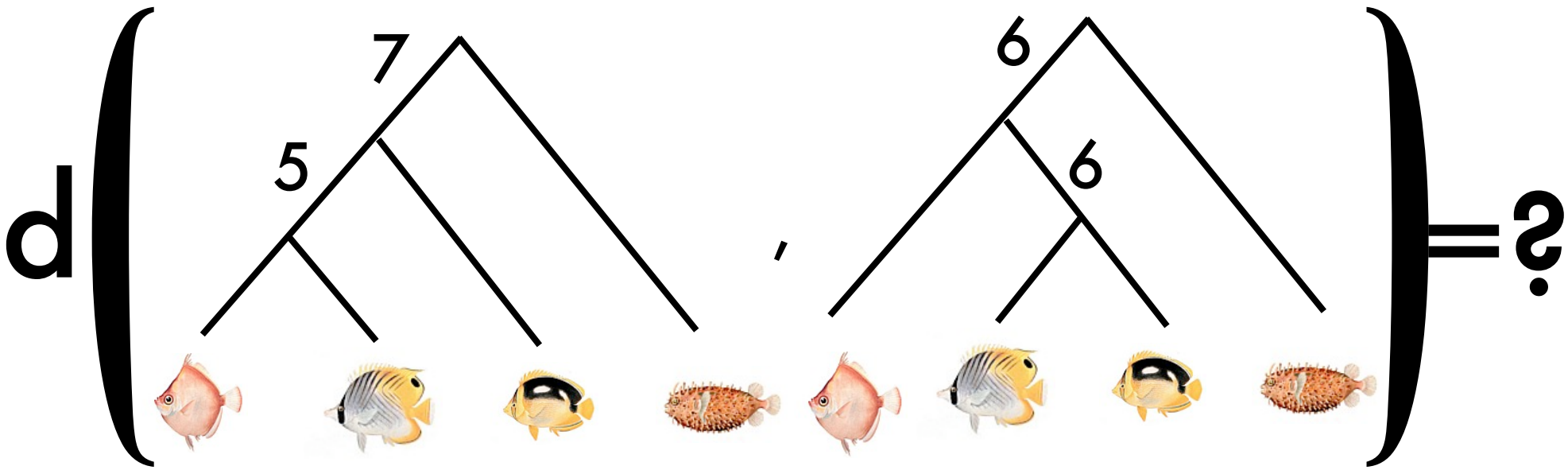
Practical Distance Computation in the Space of Phylogenetic Trees

Megan Owen
Cornell University

Problem

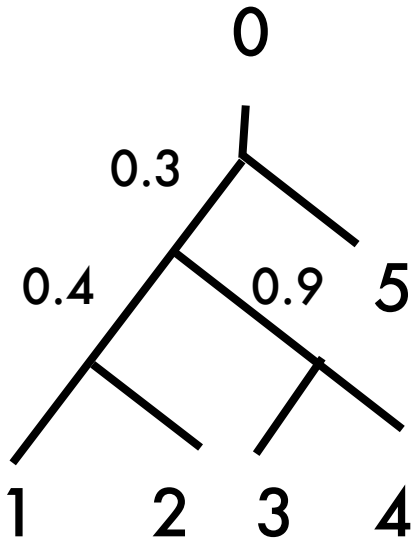


- geodesic distance introduced by Billera, Holmes, and Vogtmann in "The geometry of the space of phylogenetic trees," 2001

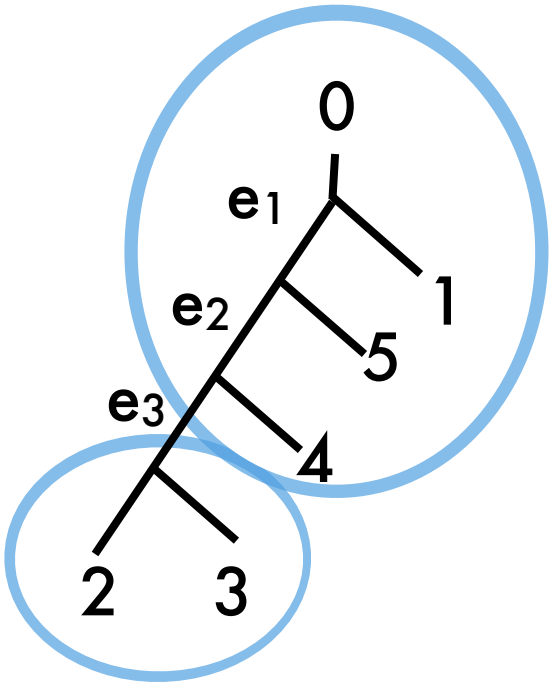
Tree Space $\mathbb{T}_n$

- $\mathbb{T}_n$ = space containing all rooted semi-labeled binary trees with n leaves and interior branch lengths ≥0
- $\mathbb{T}_n$ also contains

  degenerate trees



What is an edge?



- an interior edge partitions the set of leaves into 2:

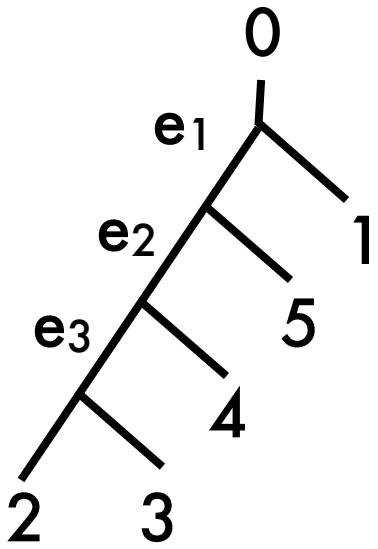
e_3 = 23 | 0145

Edge Compatibility

- $e_x = X|X'$ is *compatible* with $e_y = Y|Y'$ if both can exist in the same tree;

  more formally, if one of $X \cap Y$, $X \cap Y'$,

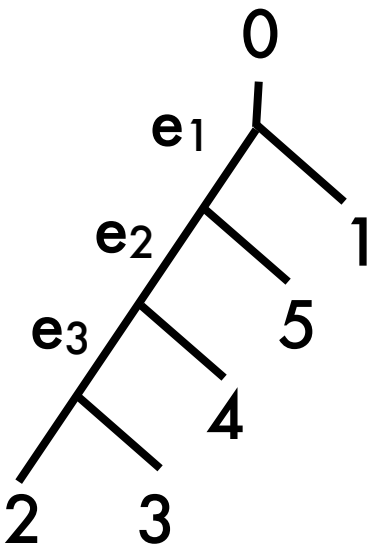
  $X' \cap Y$, or $X' \cap Y'$ is empty



ex. e_3 = 23 | 0145 is compatible with e_2 = 234 | 015

but not with f = 12 | 0345

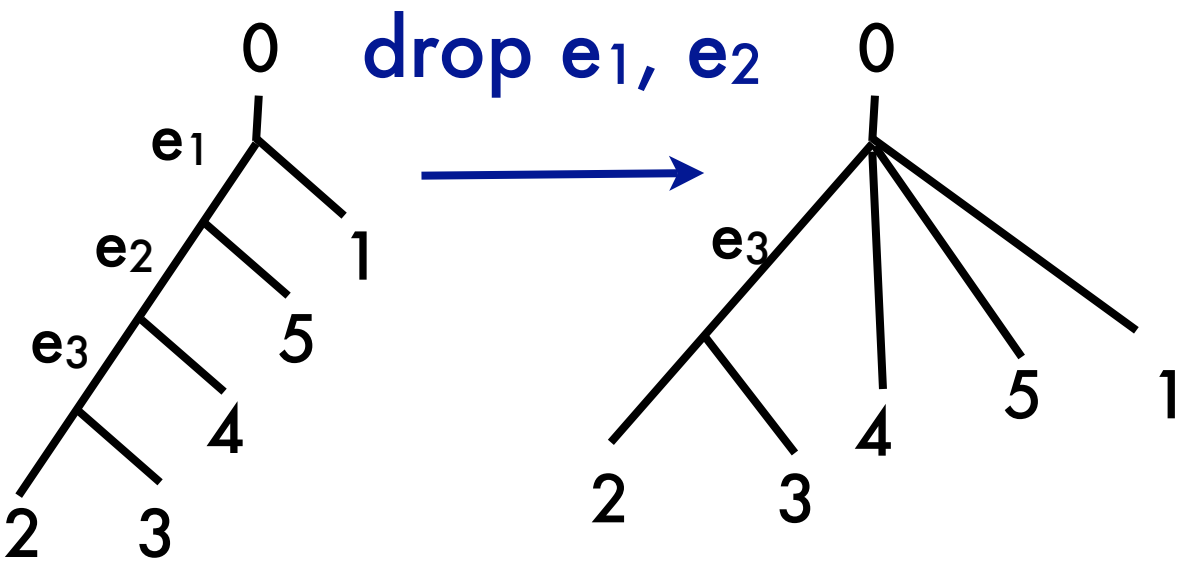
Changing trees



drop some edges

add some edges

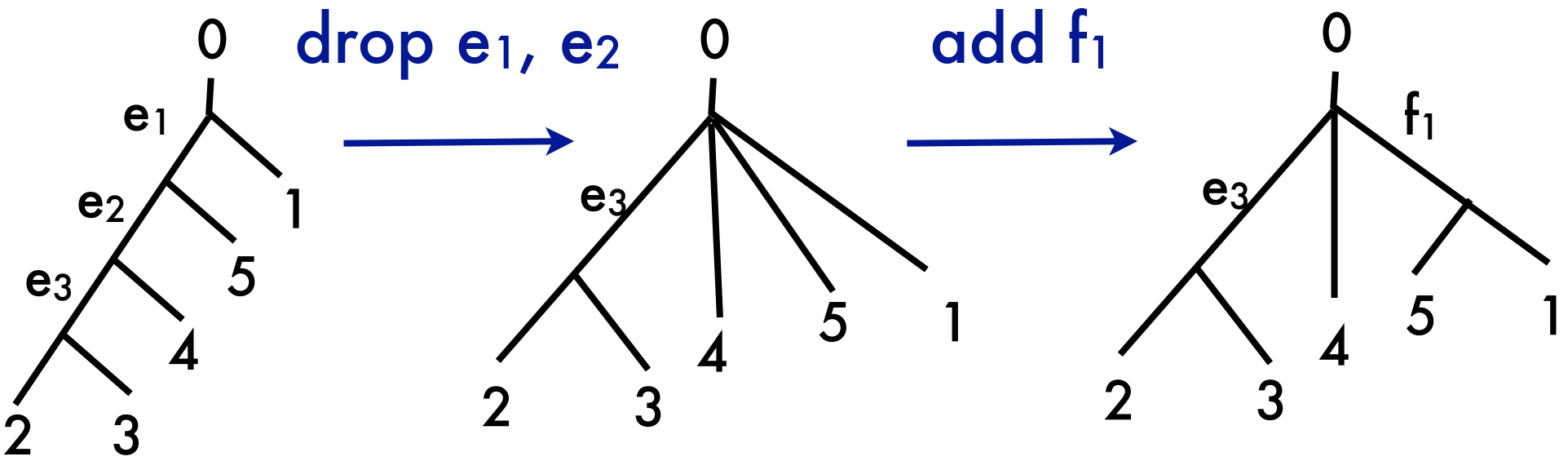
Changing trees



drop some edges

add some edges

Changing trees

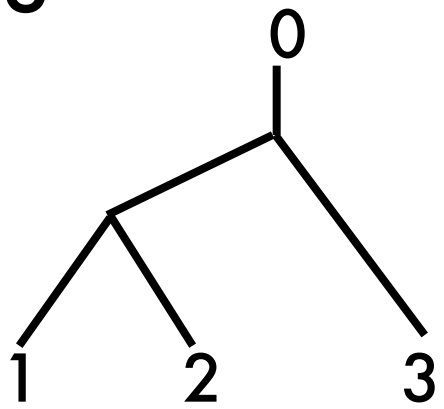


drop some edges

add some edges

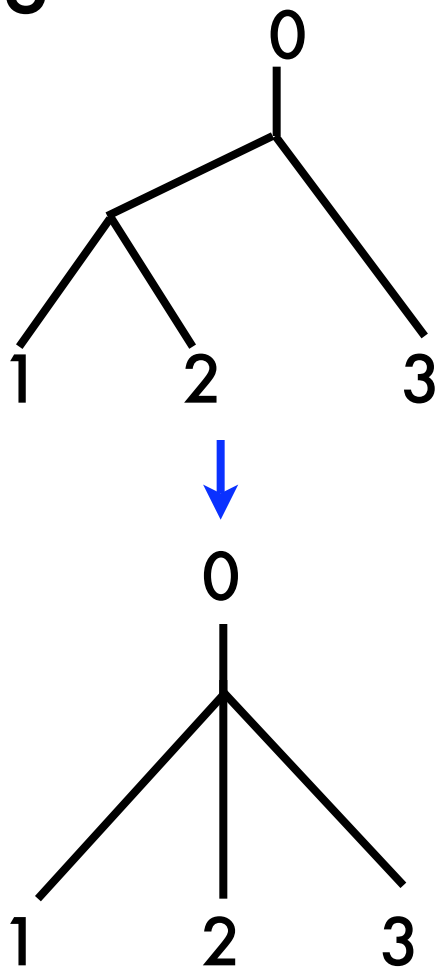
Rotations

- minimal change is drop one edge & add one edge = *rotation*



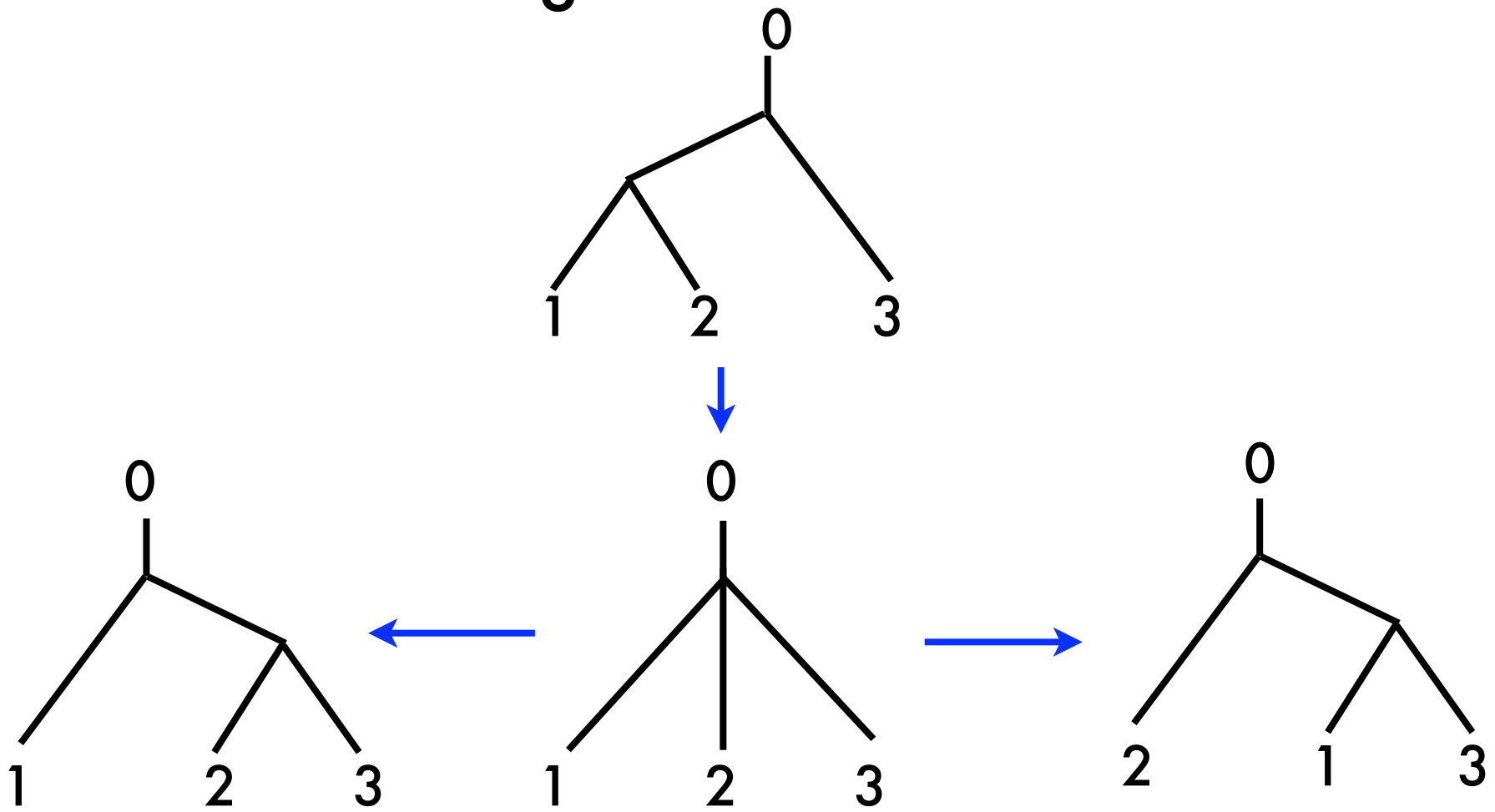
Rotations

- minimal change is drop one edge & add one edge = *rotation*

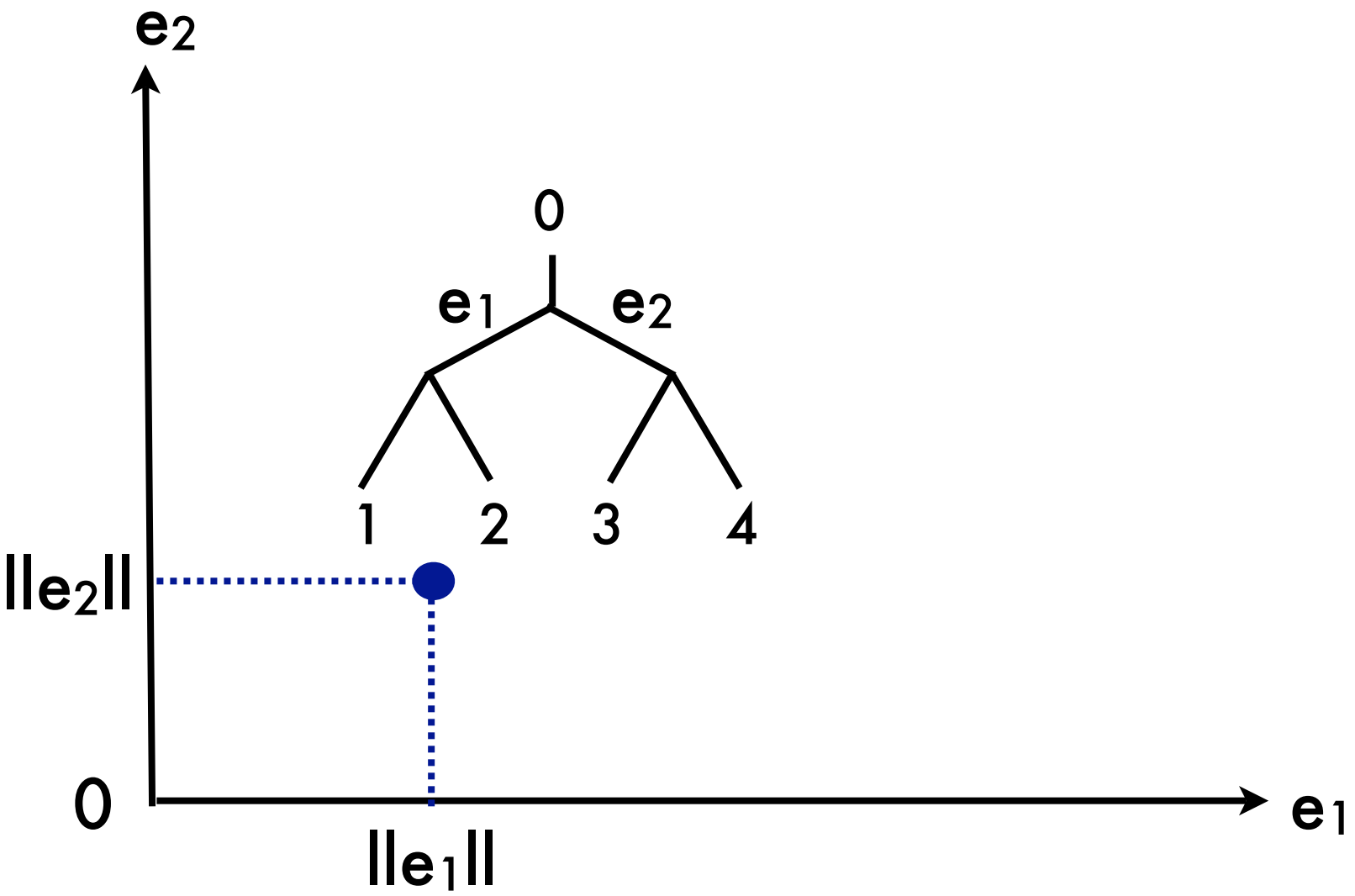


Rotations

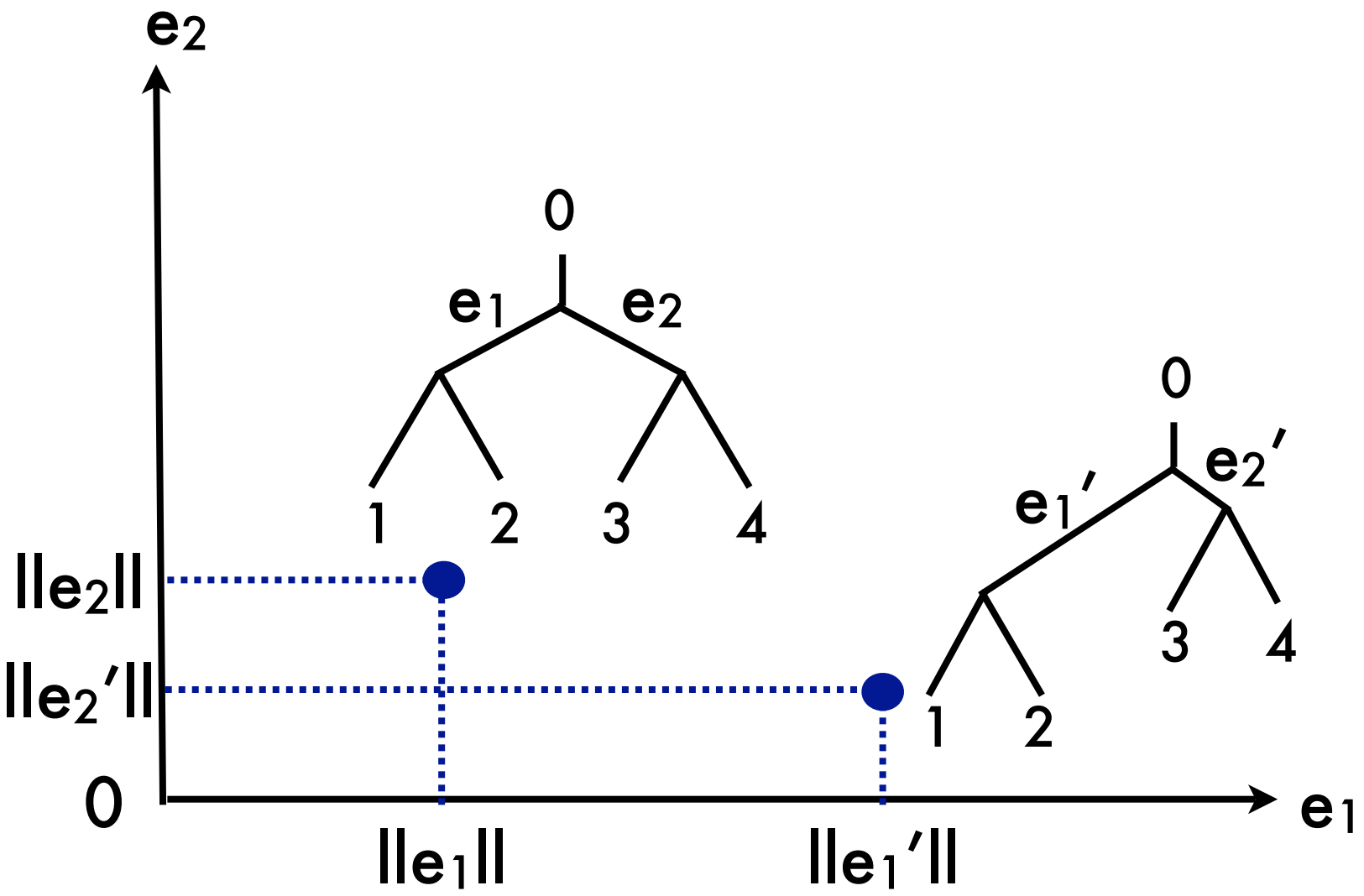
- minimal change is drop one edge & add one edge = *rotation*



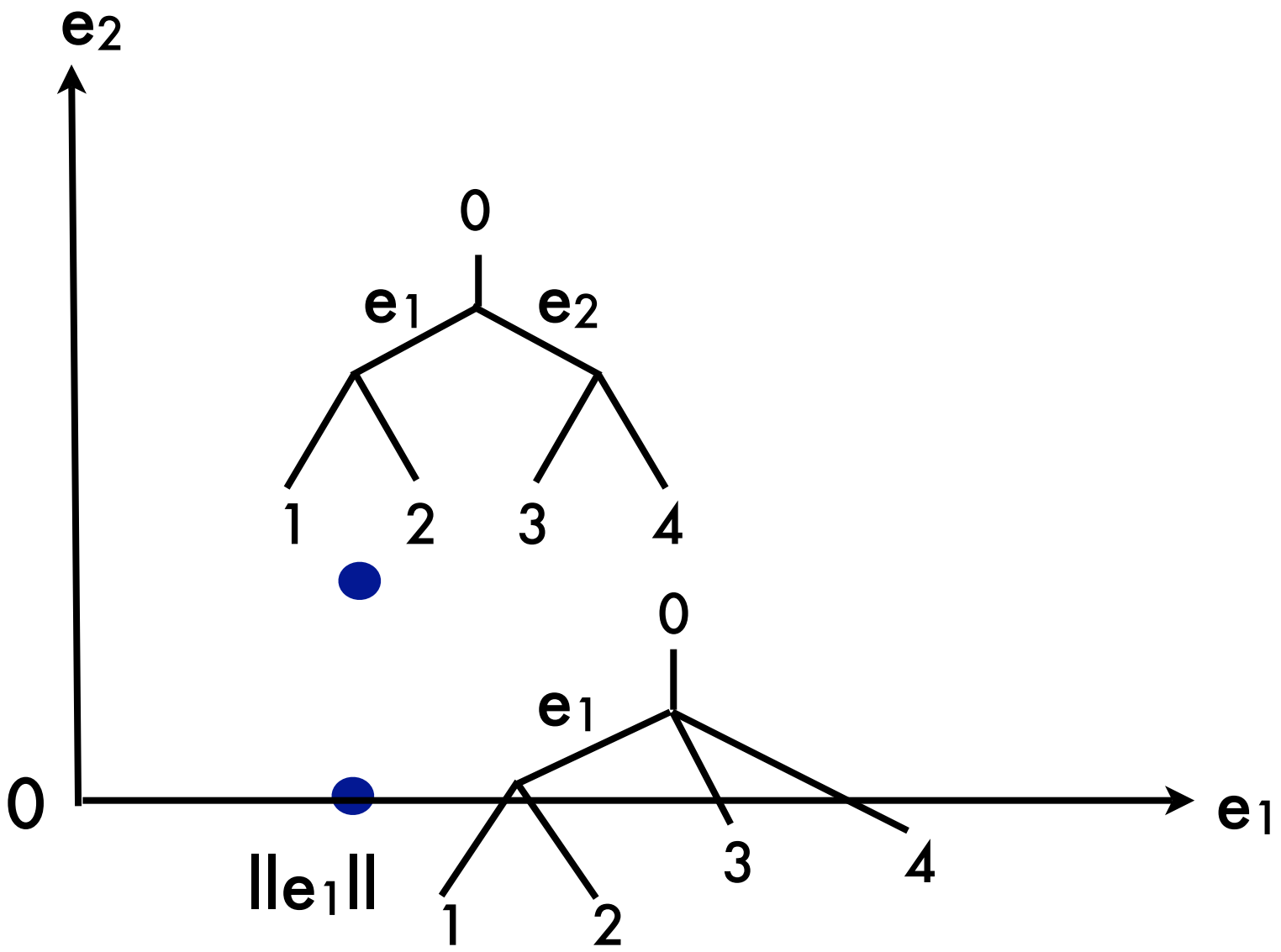
Orthants



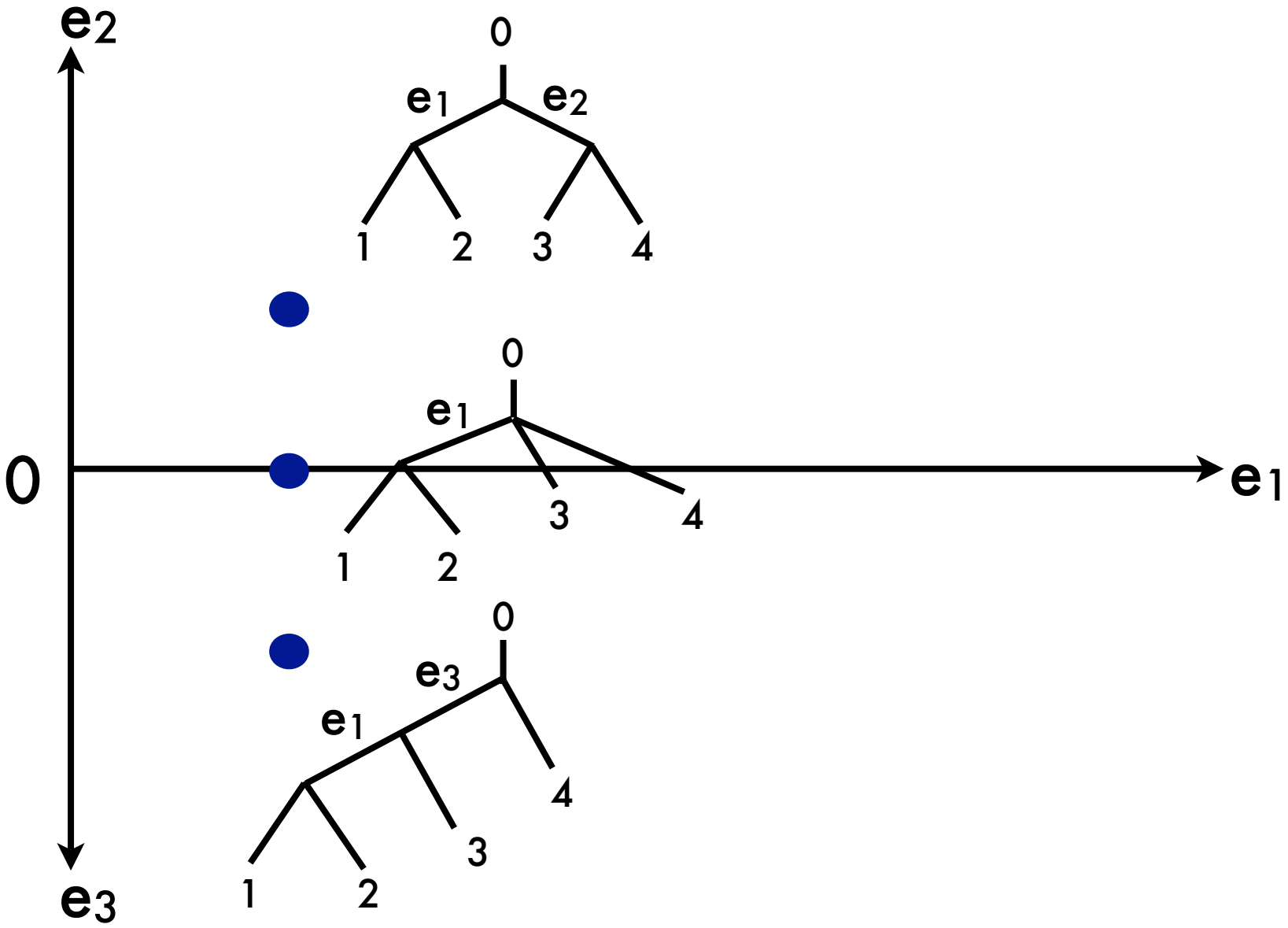
Orthants



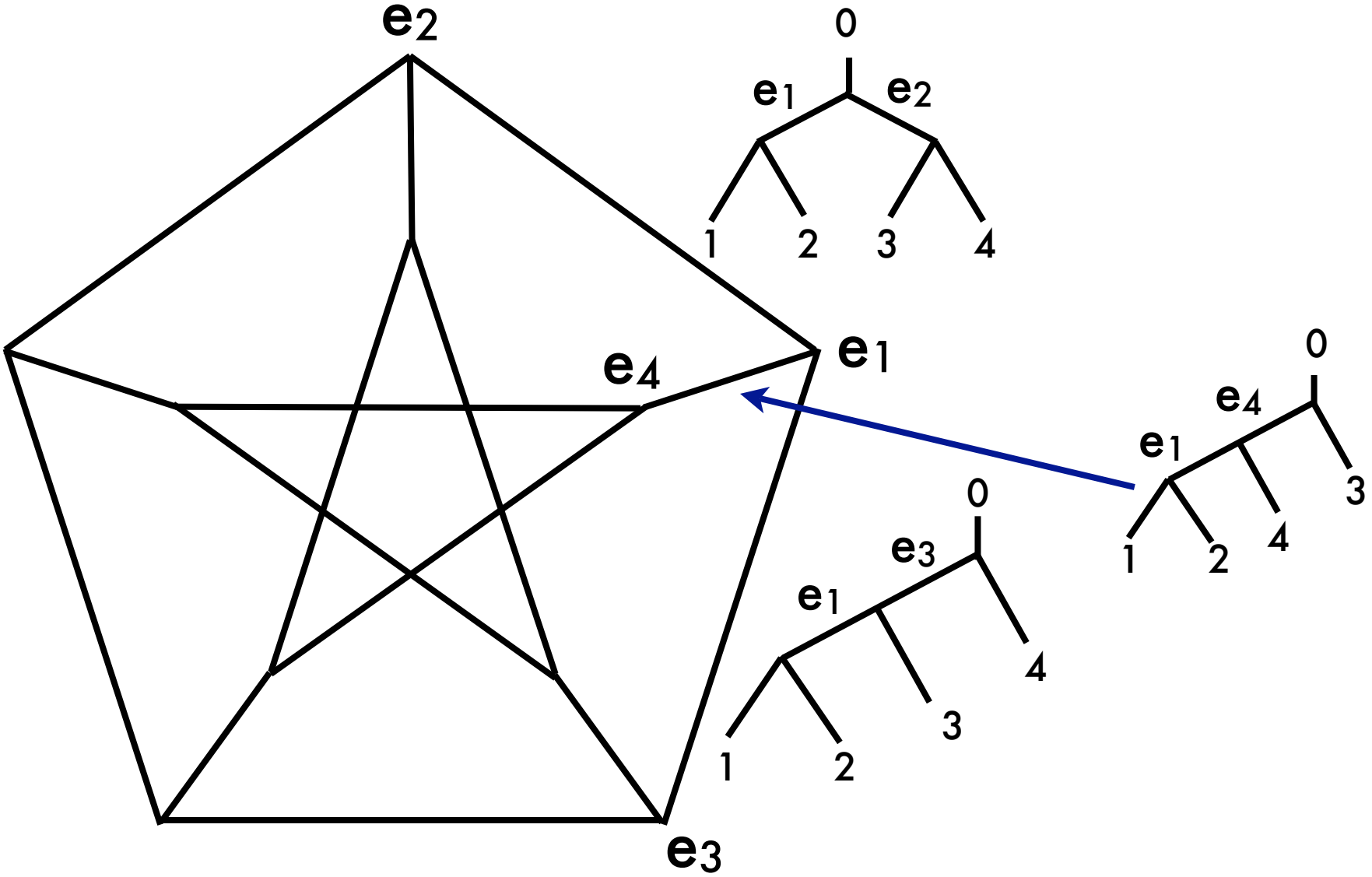
Orthants



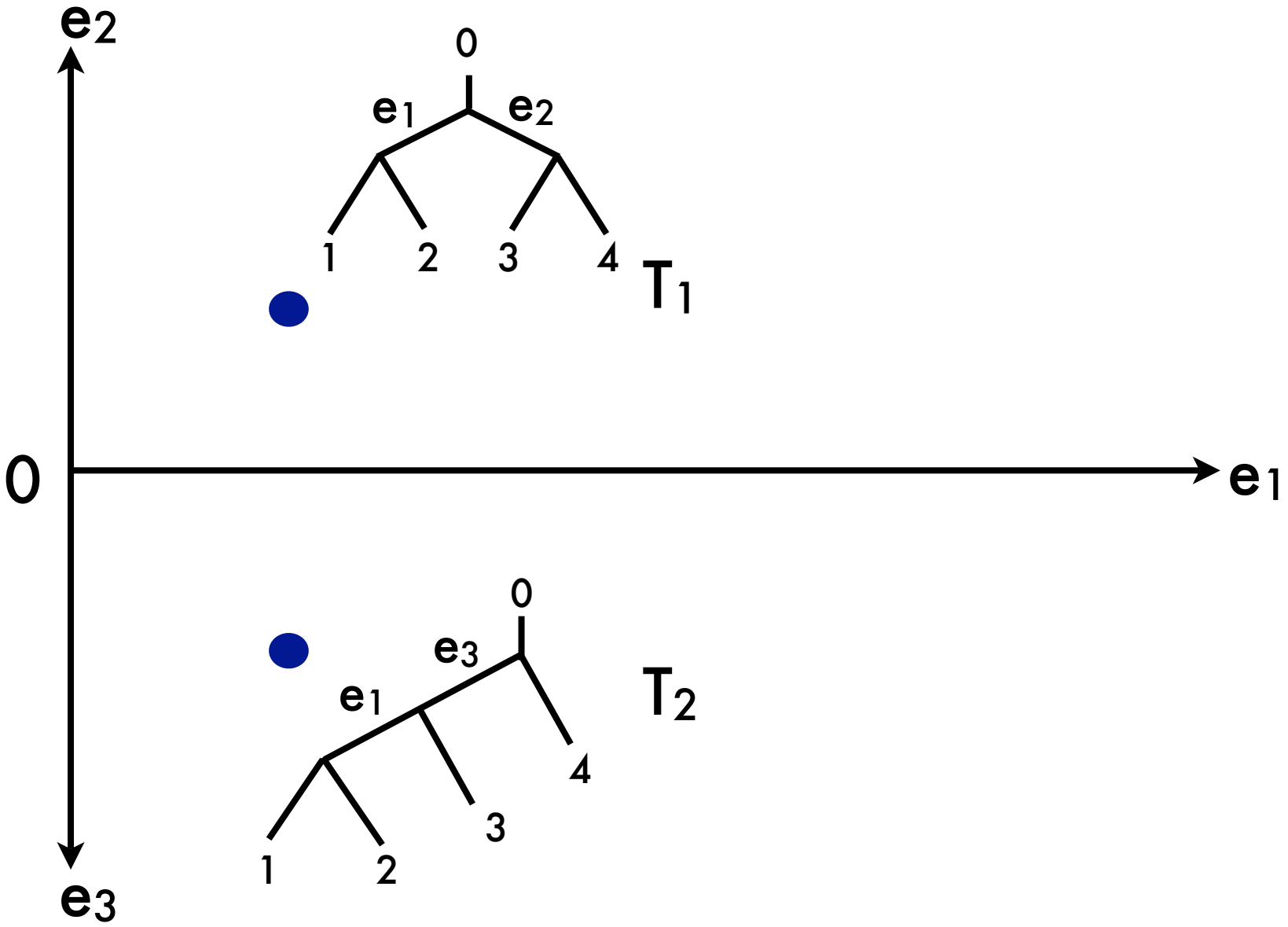
Orthants



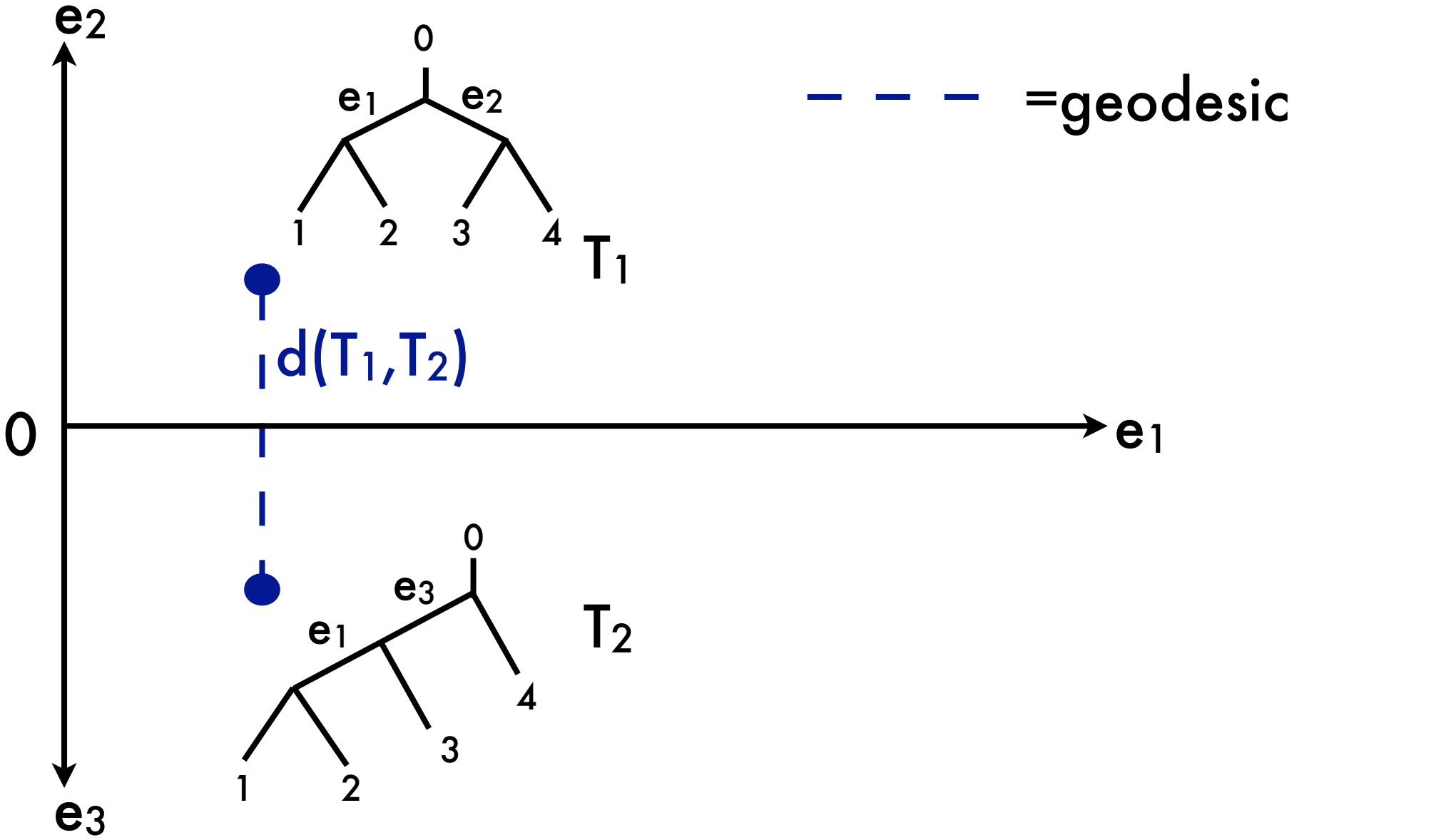
Structure of $\mathbb{T}_4$



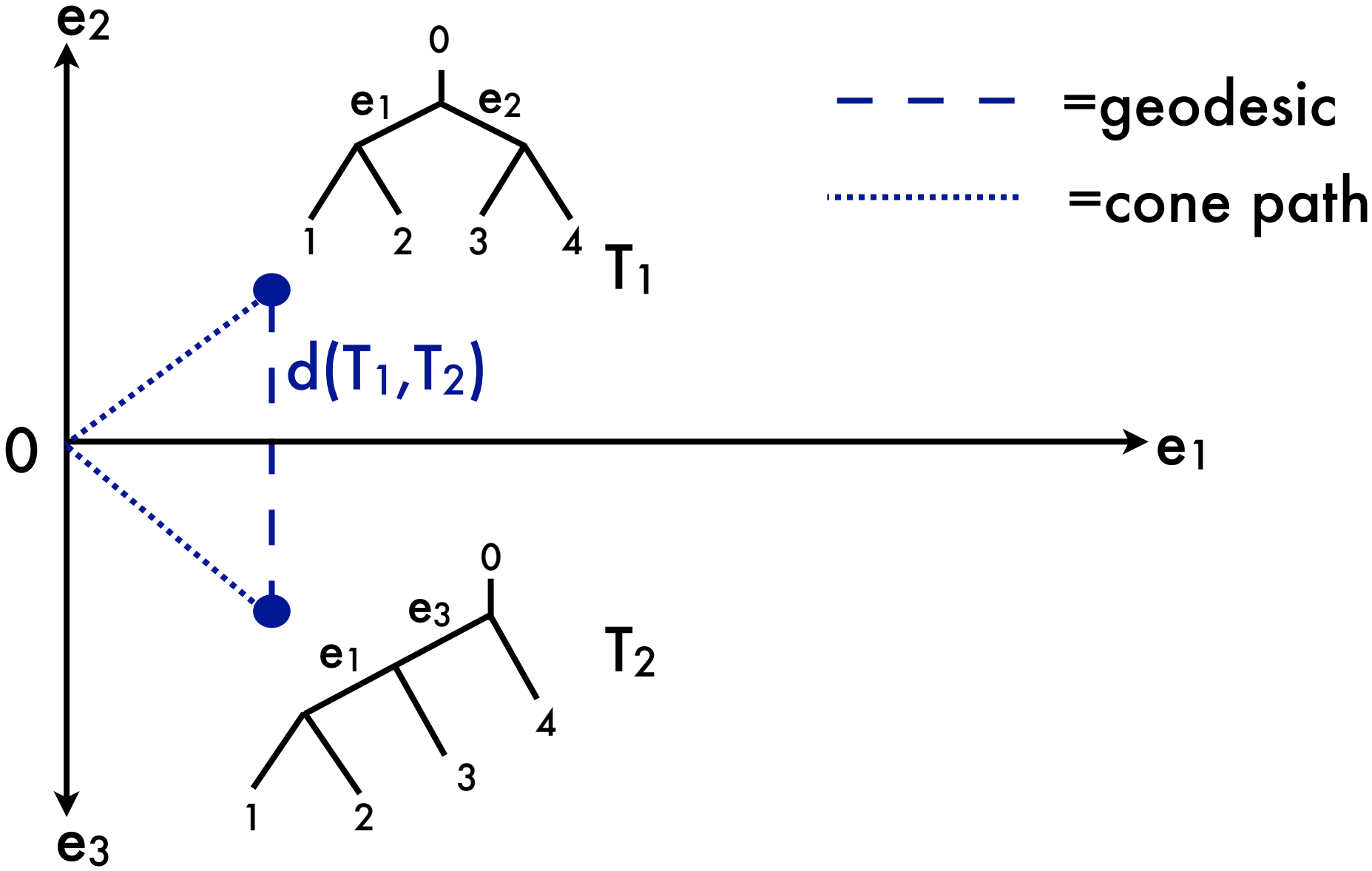
Geodesic Distance



Geodesic Distance



Geodesic Distance



Properties of $\mathbb{T}_n$

- CAT(0) space

  $\Rightarrow$ unique geodesic

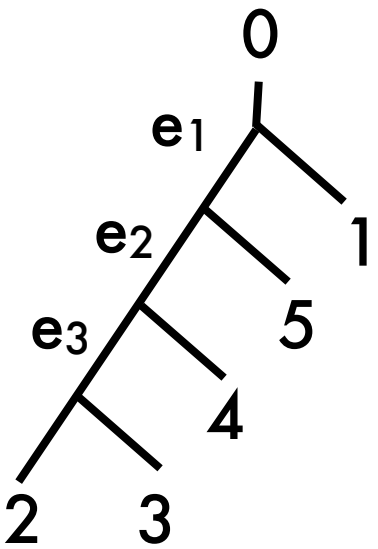
- geodesic = shortest path between two points

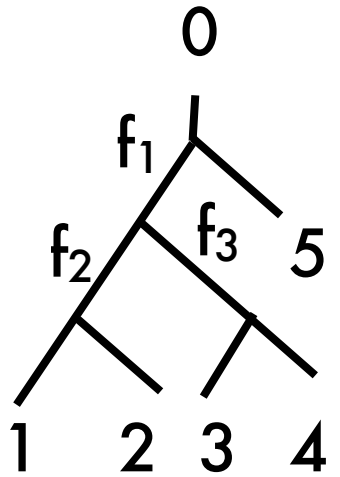
Main Question

- Can we find an efficient algorithm to compute the geodesic between two trees in $\mathbb{T}_n$?

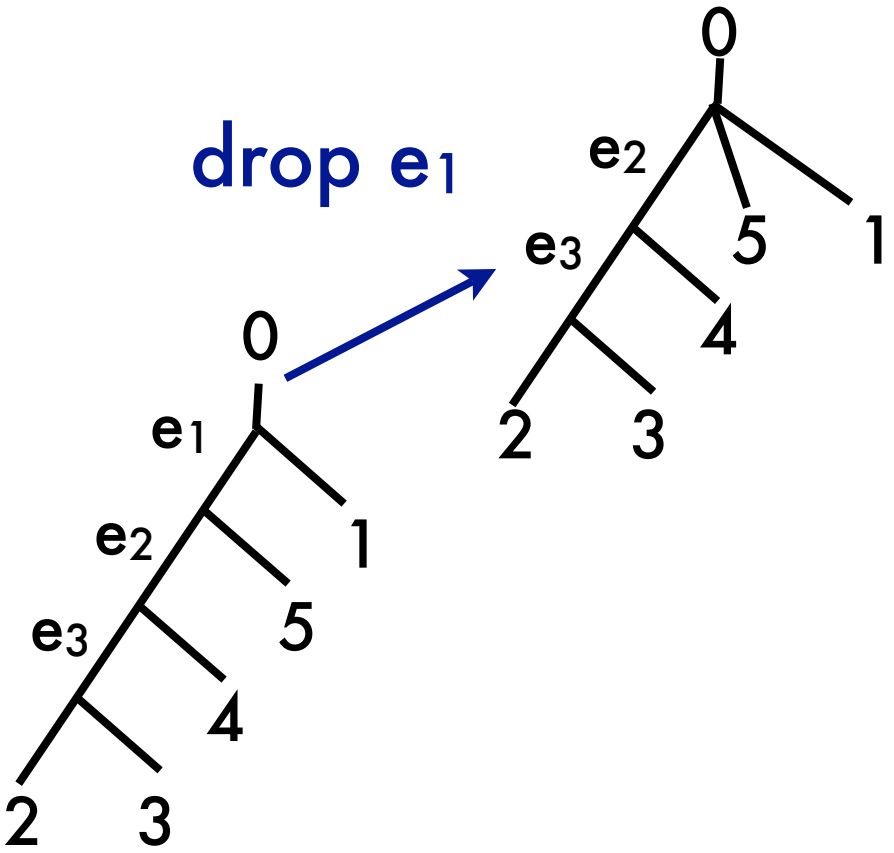
- for now, assume the two trees have no shared edges

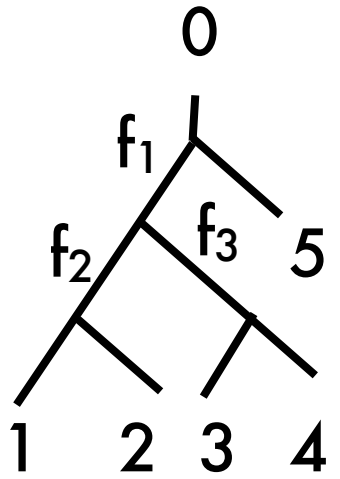
Path Spaces



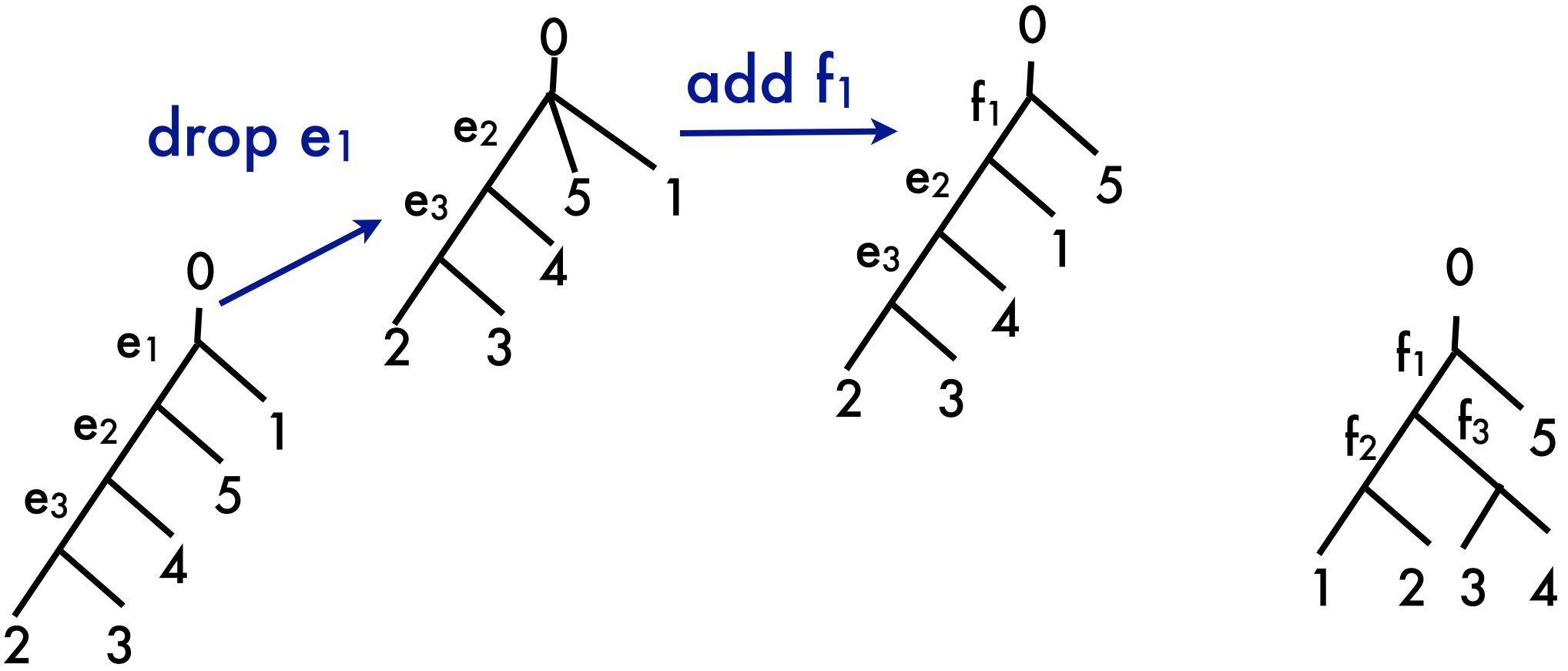


Path Spaces

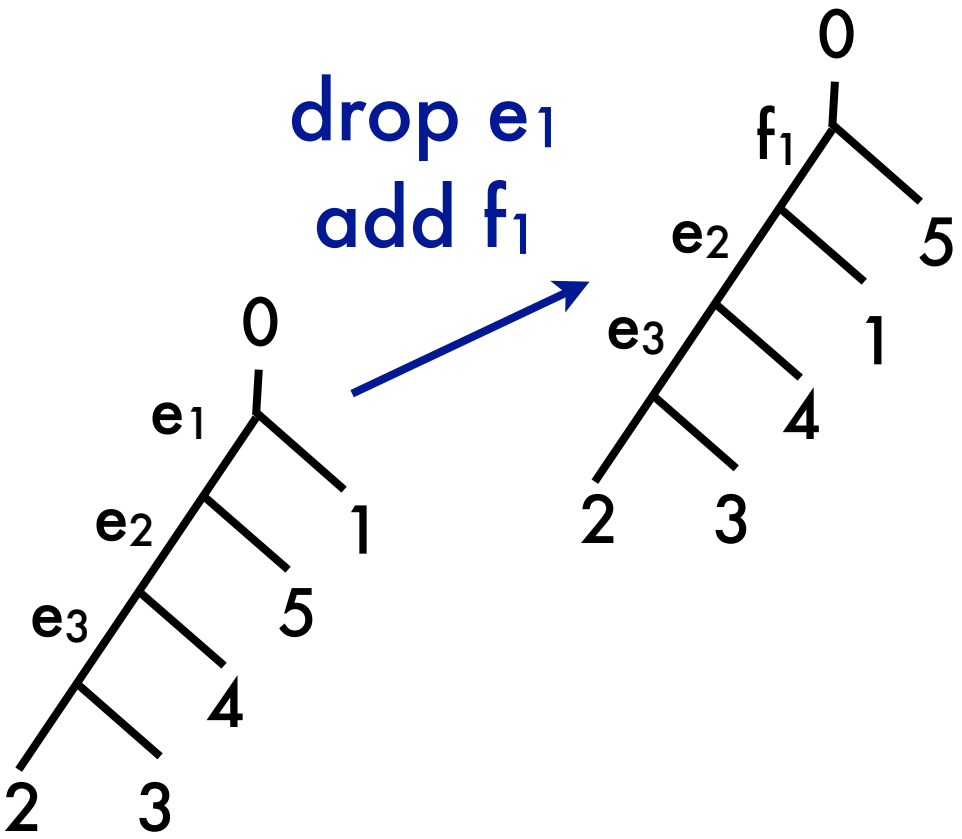


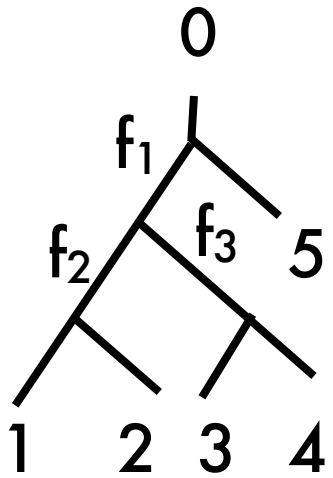


Path Spaces

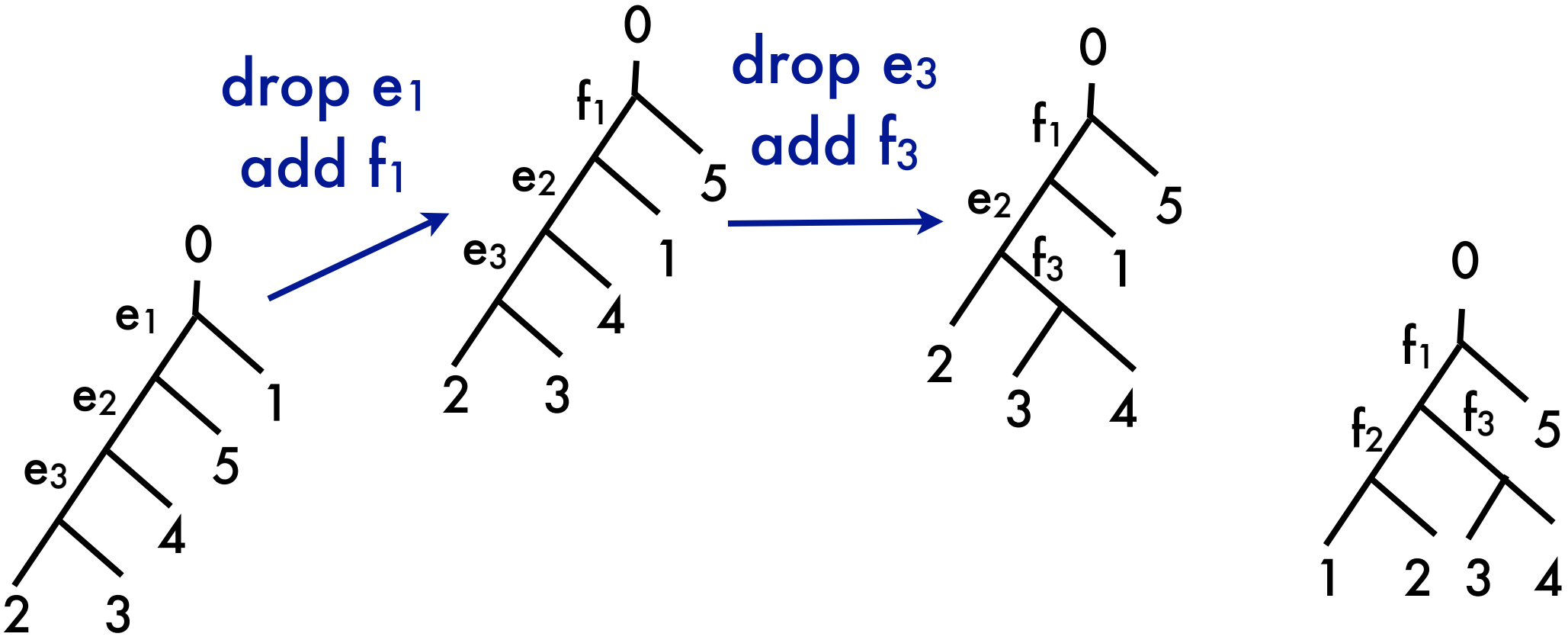

drop e_1
add f_1
0
e_1
e_2
e_3
1
5
4
2
3
0
e_2
e_3
5
1
4
2
3
0
f_1
e_2
5
e_3
1
4
2
3
0
f_1
f_2
f_3
5
1
2
3
4

Path Spaces

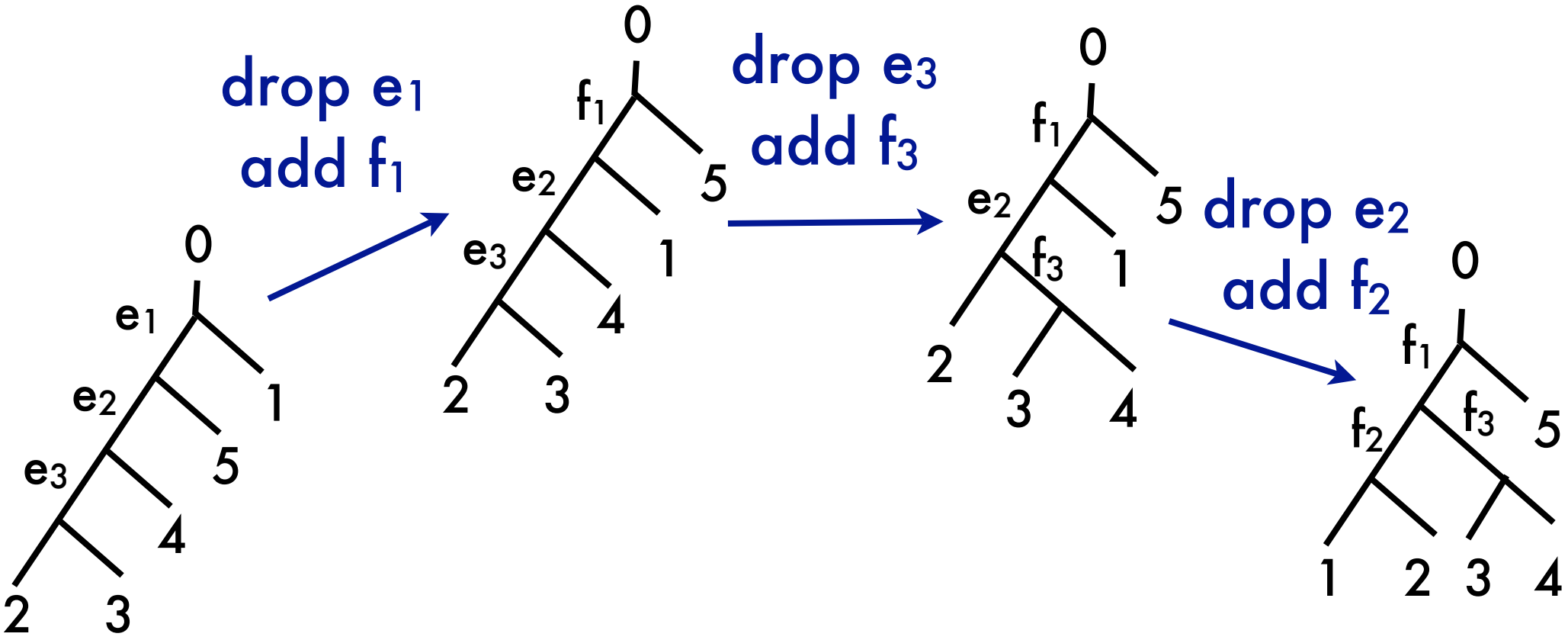




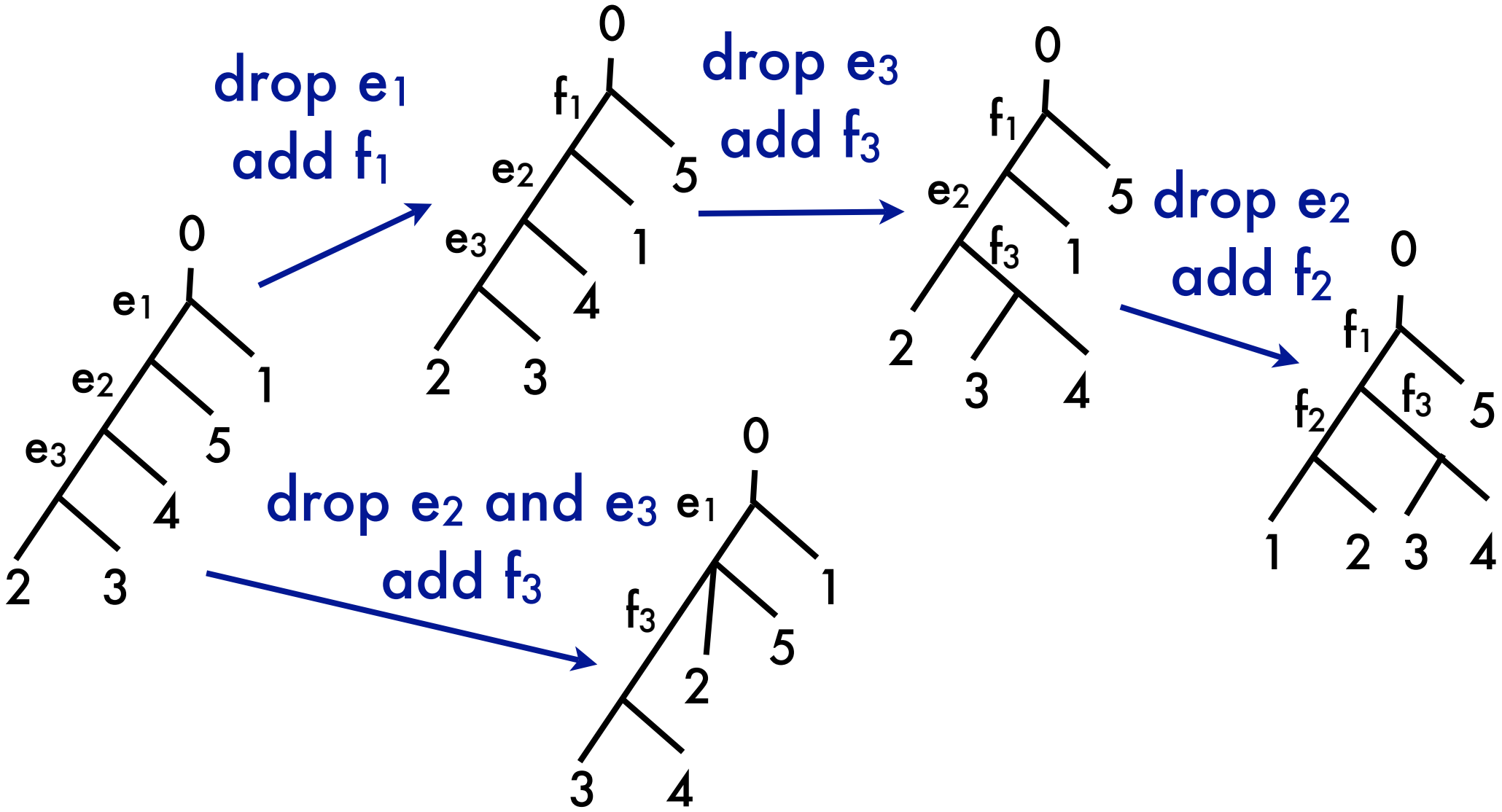
Path Spaces


drop e_1
add f_1
drop e_3
add f_3
0
e_1
e_2
e_3
1
5
4
2
3
0
f_1
e_2
e_3
5
1
4
2
3
0
f_1
e_2
f_3
5
1
2
3
4
0
f_1
f_2
f_3
5
1
2
3
4

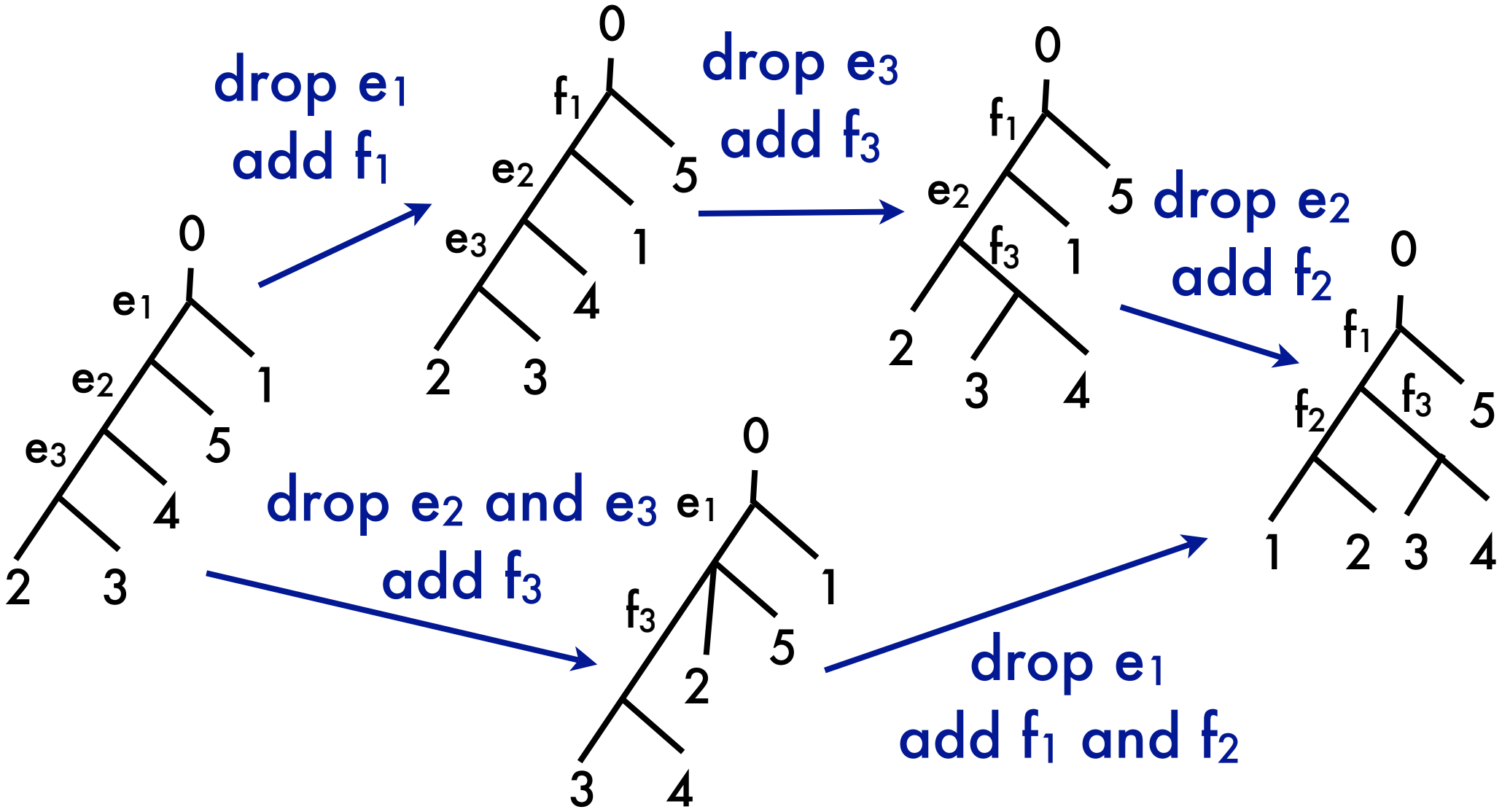
Path Spaces


drop e_1
add f_1
drop e_3
add f_3
drop e_2
add f_2
0
e_1
e_2
e_3
1
5
4
2
3
0
f_1
e_2
e_3
5
1
4
2
3
0
f_1
e_2
f_3
5
1
2
3
4
0
f_1
f_2
f_3
5
1
2
3
4

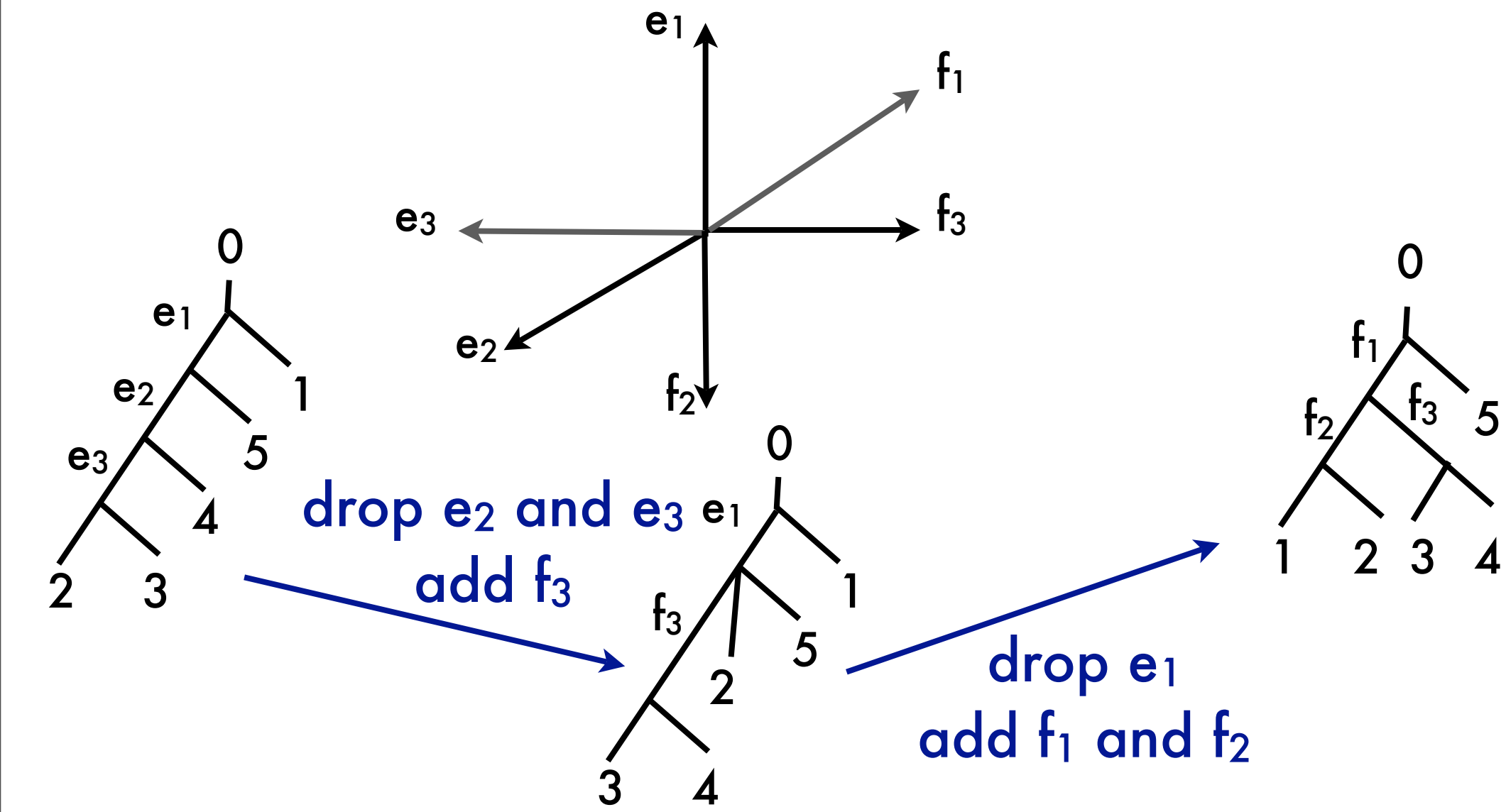
Path Spaces


drop e_1
add f_1
drop e_3
add f_3
drop e_2
add f_2
drop e_2 and e_3
add f_3
0
e_1
e_2
e_3
1
5
4
2
3
f_1
f_2
f_3

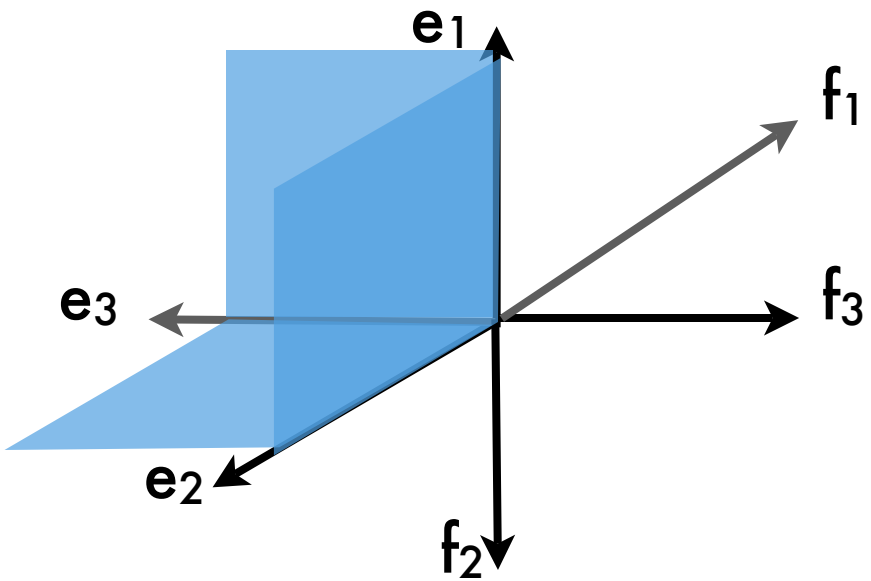
Path Spaces

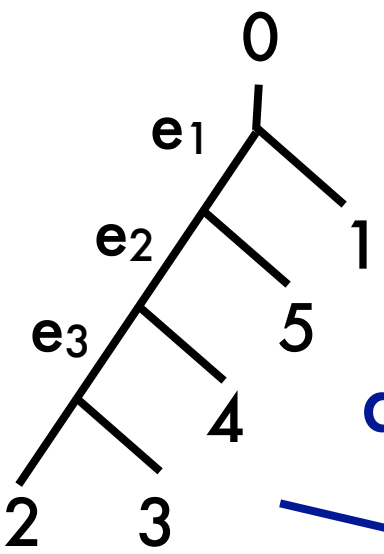

drop e_1
add f_1
drop e_3
add f_3
drop e_2
add f_2
drop e_2 and e_3
add f_3
drop e_1
add f_1 and f_2
0
e_1
e_2
e_3
1
5
4
2
3
f_1
f_3
f_2

Path Spaces

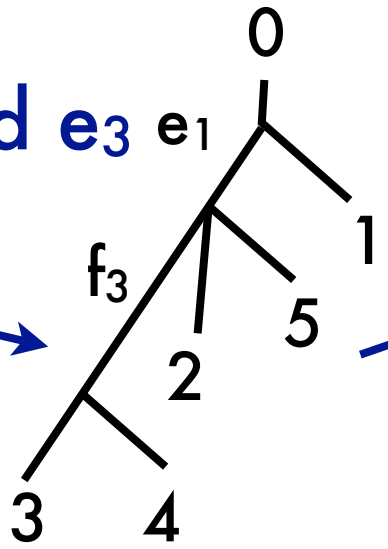

e1
f1
e3
f3
e2
f2
0
e1
e2
1
e3
5
4
2
3
drop e2 and e3
add f3
0
e1
1
f3
5
2
3
4
drop e1
add f1 and f2
0
f1
f2
f3
5
1
2
3
4

Path Spaces

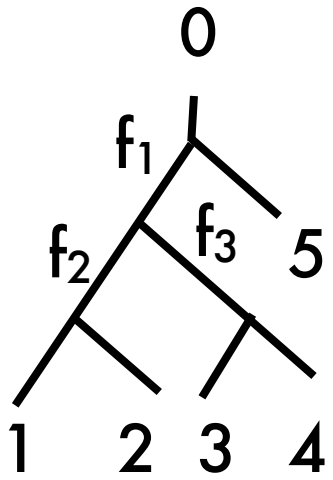




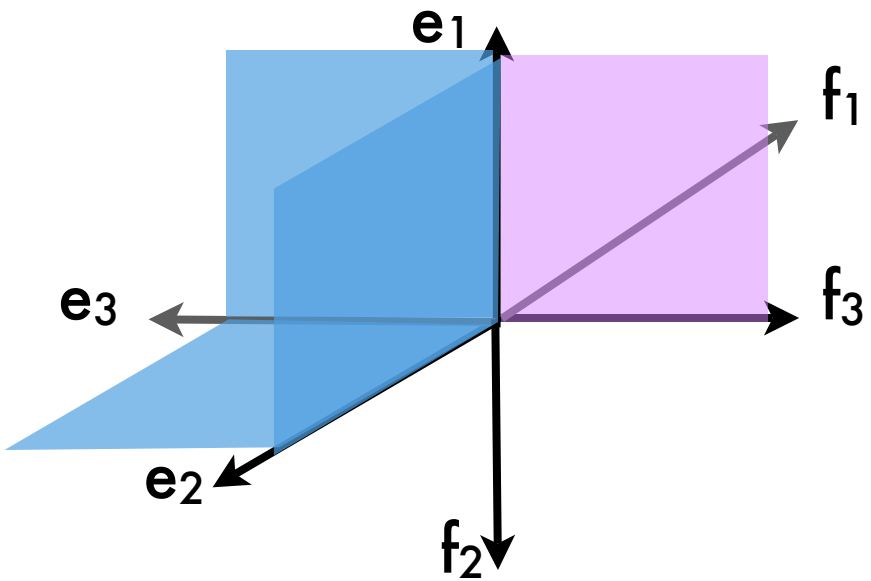
drop e_2 and e_3
add f_3

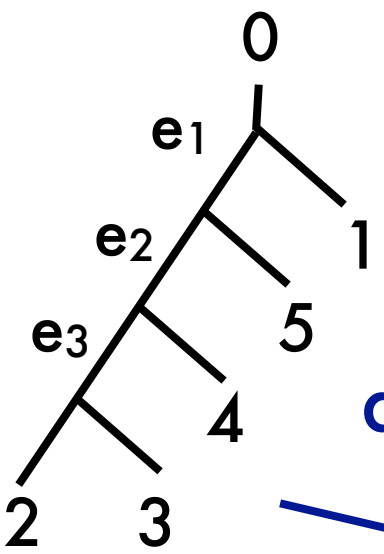


drop e_1
add f_1 and f_2

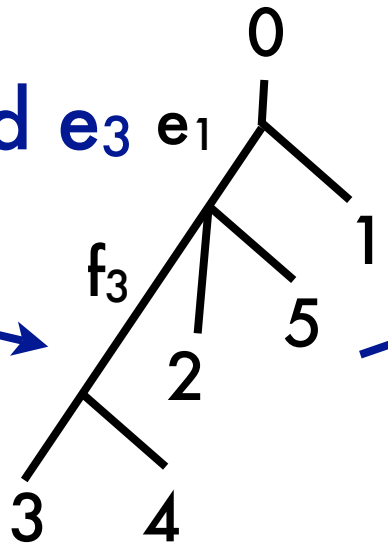


Path Spaces

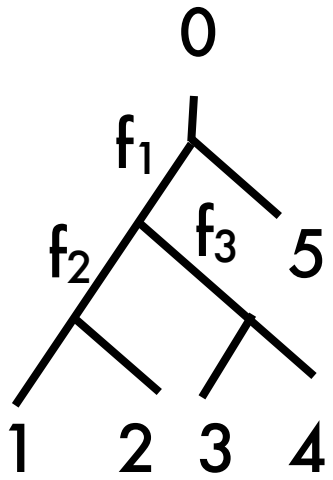




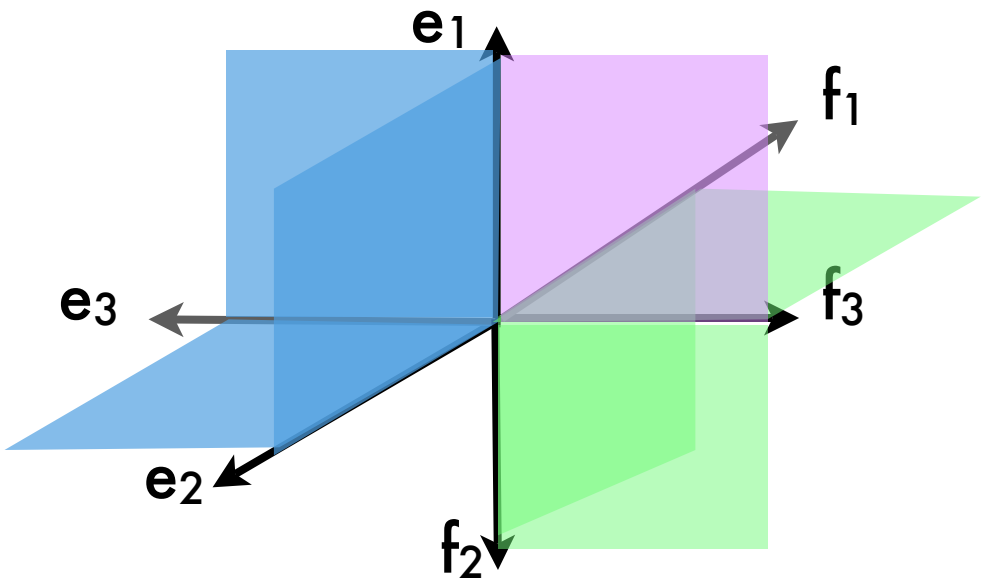
drop e_2 and e_3
add f_3

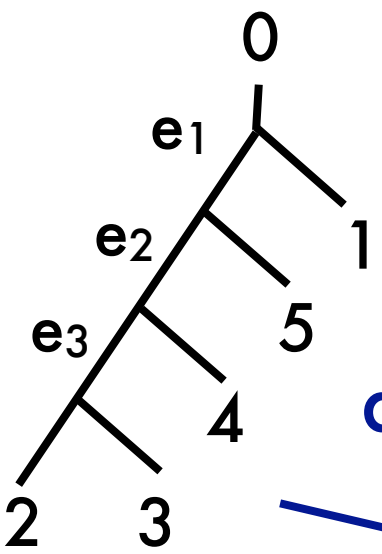


drop e_1
add f_1 and f_2

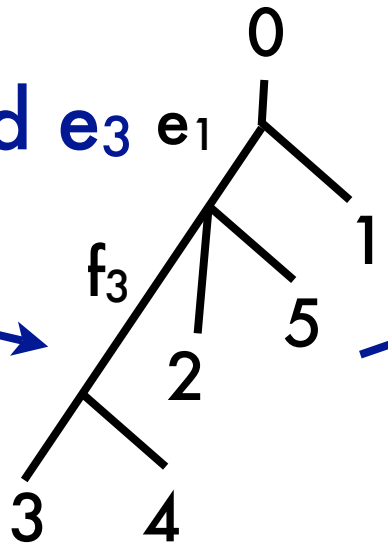


Path Spaces

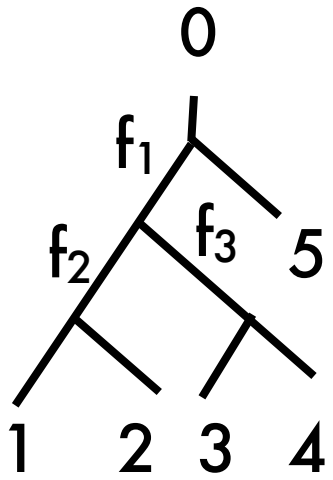




drop e_2 and e_3
add f_3



drop e_1
add f_1 and f_2



Path Spaces

- Billera, Holmes and Vogtmann (2001) showed the geodesic is contained in a path space.

Plan for Algorithm

Plan for Algorithm

1. Find simple enumeration of all *maximal* path spaces. (at least one contains the geodesic)
2. $\forall$ maximal path space, show how to compute the local geodesic.
3. Combine 1. and 2. using dynamic programming techniques.

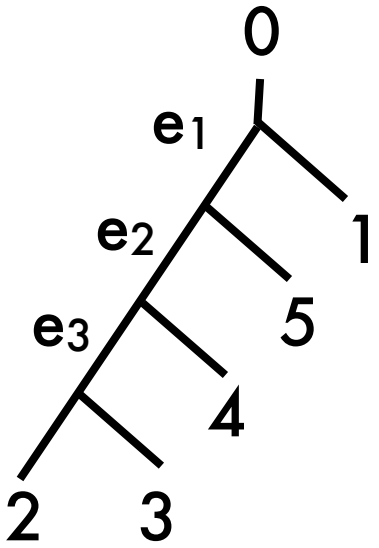
1: Max path spaces

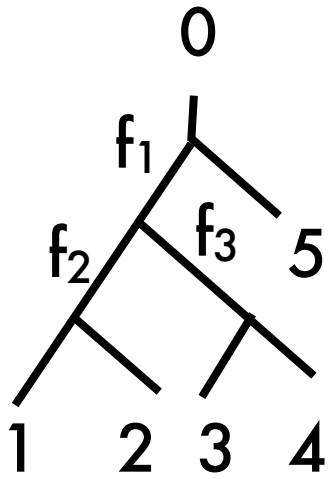
- intuitively, orthants in a max. path space are as large as possible

  $\Rightarrow$ at each step:

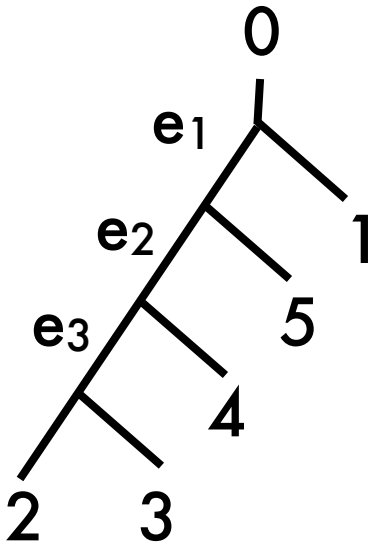
  - drop as few edges as possible
  - add as many edges as possible
  - don't add edges not in the target tree

1: Partition Poset - P

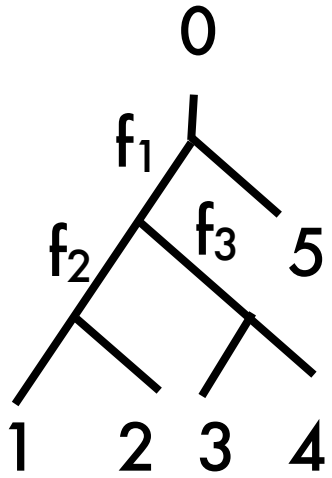




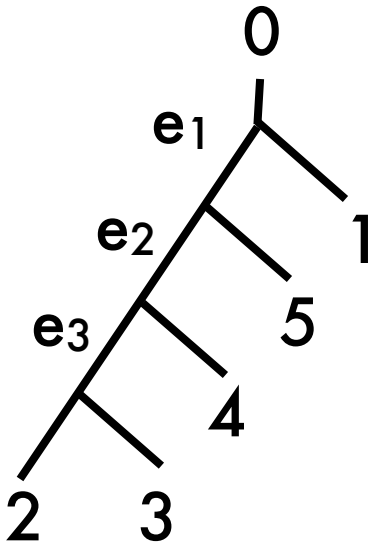
1: Partition Poset - P



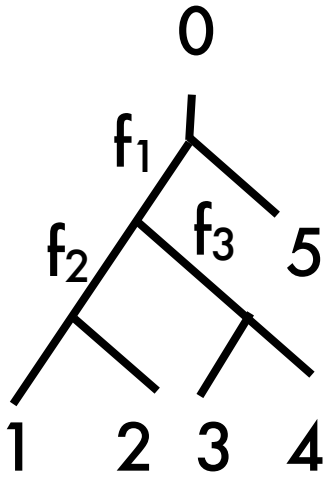
- f_1 incompatible with e_1



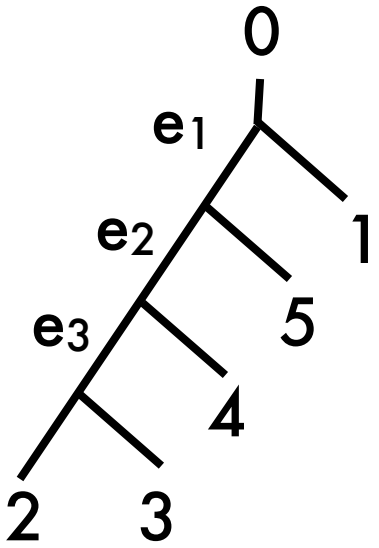
1: Partition Poset - P



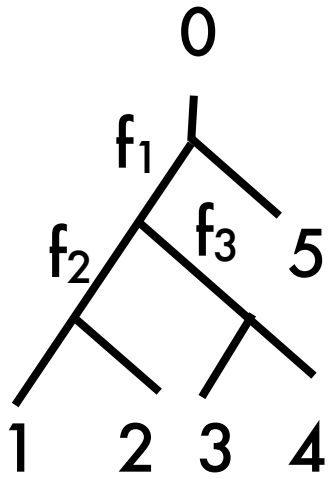
- f_1 incompatible with e_1
- f_2 incompatible with e_1, e_2, e_3



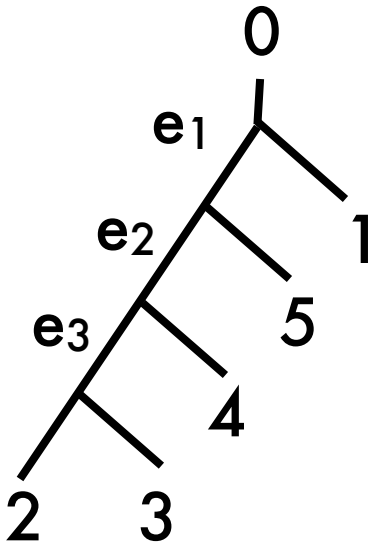
1: Partition Poset - P



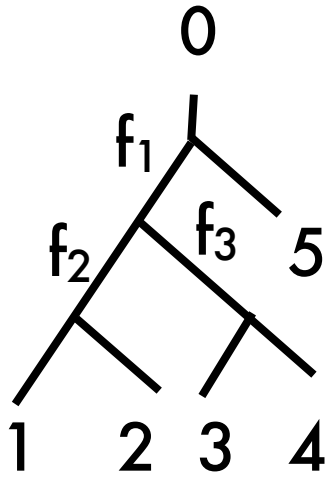
- f_1 incompatible with e_1
- f_2 incompatible with e_1, e_2, e_3
- f_3 incompatible with e_3



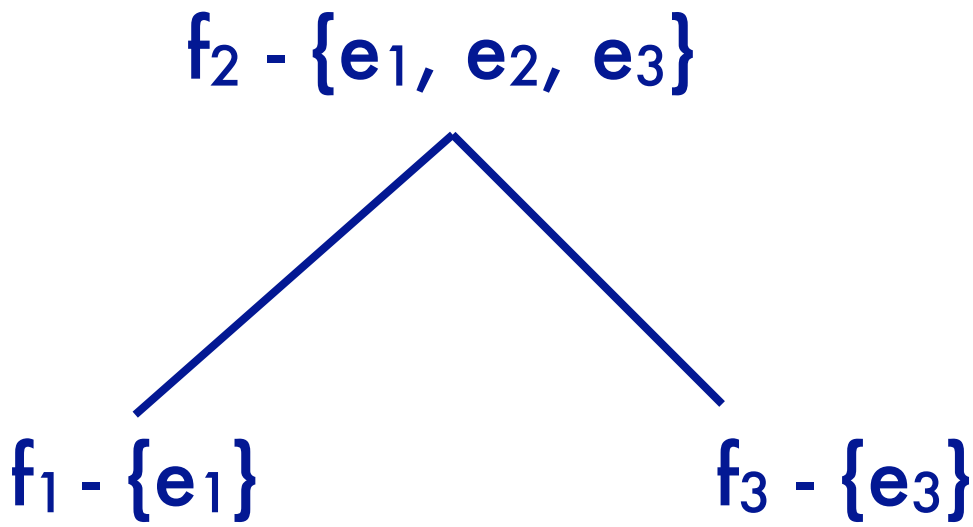
1: Partition Poset - P



- f_1 incompatible with e_1
- f_2 incompatible with e_1, e_2, e_3
- f_3 incompatible with e_3



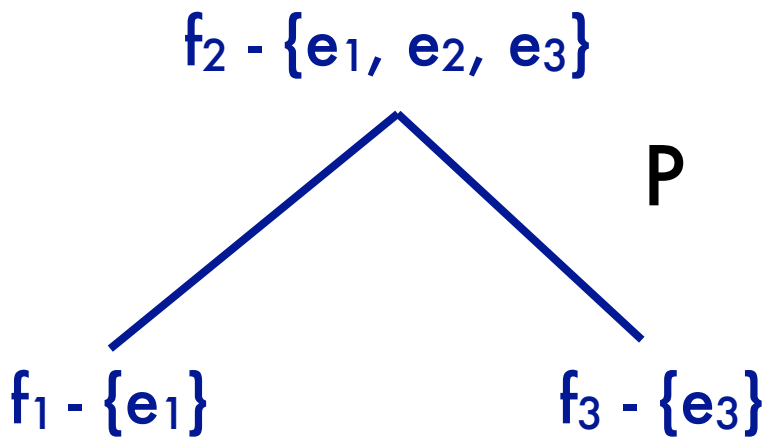
P



1: Back to Max Path Space

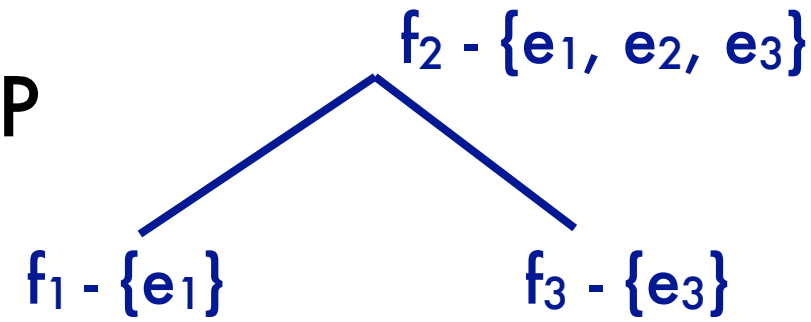
- max. path space iff at each step:
  - add a min element, f, from P\{edges already added}
  - drop all remaining edges incompatible with f
  - add other edges

    from T_f if possible



1: Closure

- A = set of edges in T_f
- closure of A = all edges from T_f that can be added when we drop the edges in T_e necessary to bring in A



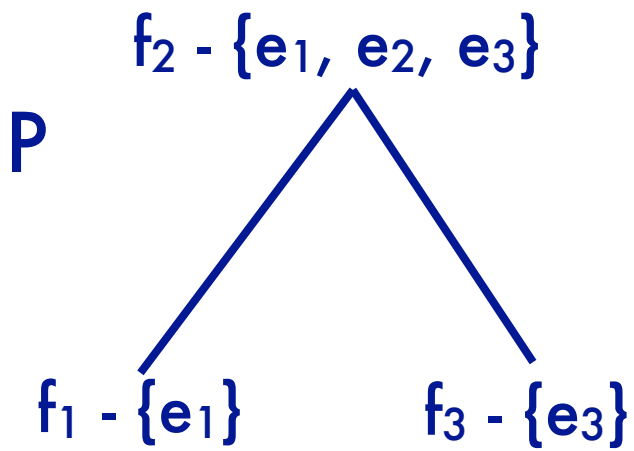
- more formally:

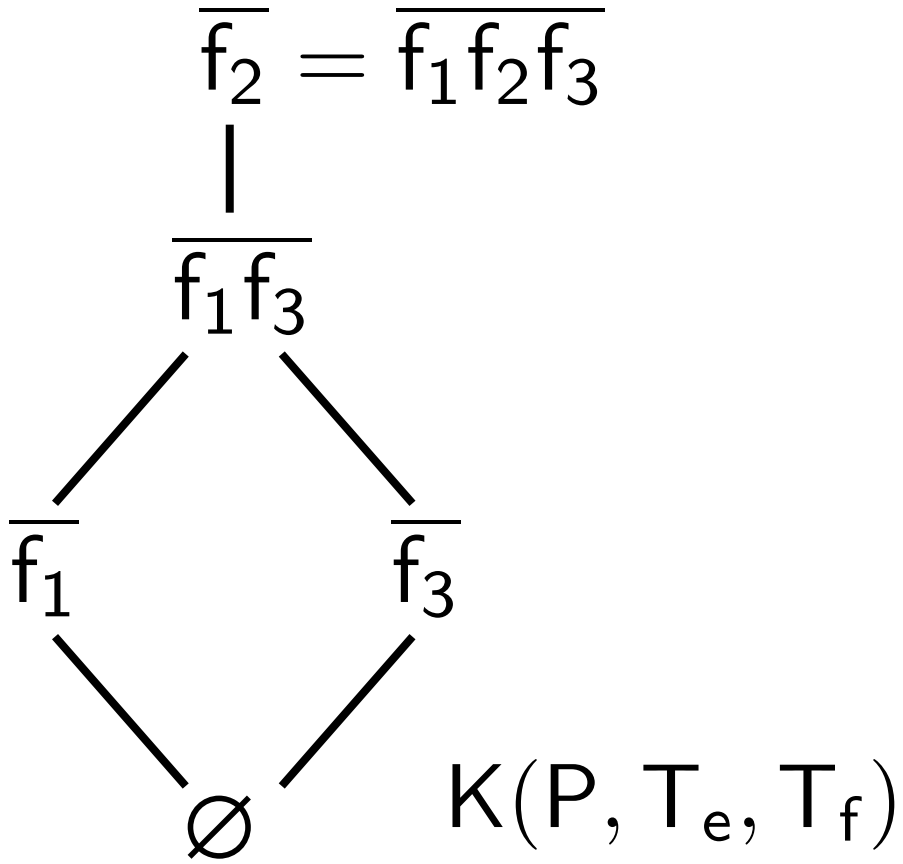
$$\overline{A} = \{f \in E_f : X_{T_e}(f) \subseteq X_{T_e}(A)\}$$

where $X_{T_e}(B)$= all the edges in T_e that are incompatible with at least one edge in B

1: Path Poset - $K(P, T_e, T_f)$

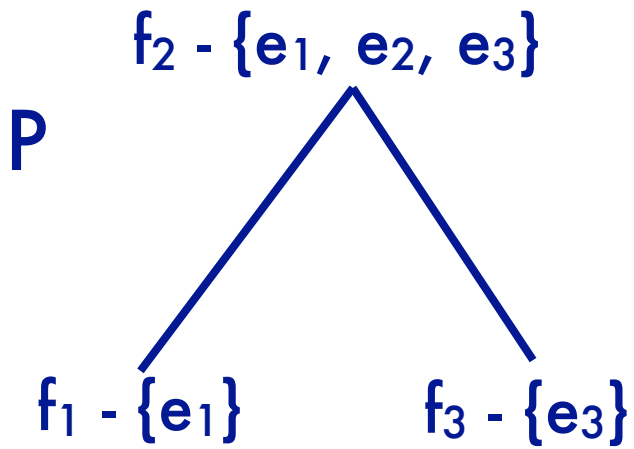
- path poset = closed sets of $E(T_f)$ ordered by inclusion

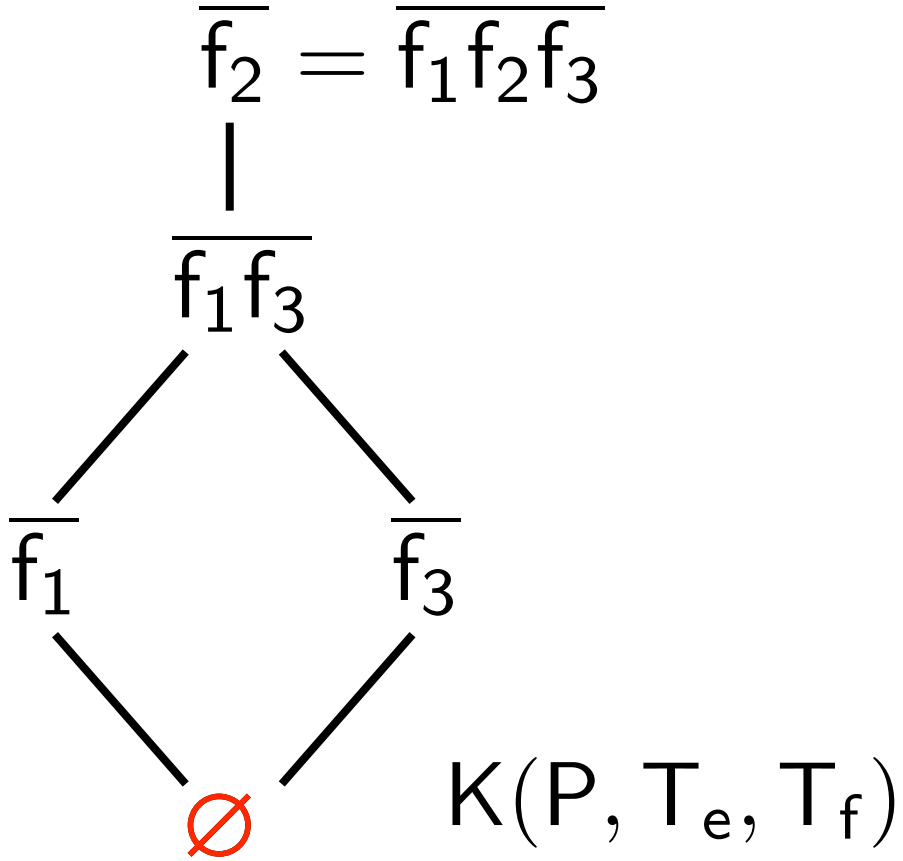




1: Path Poset - $K(P, T_e, T_f)$

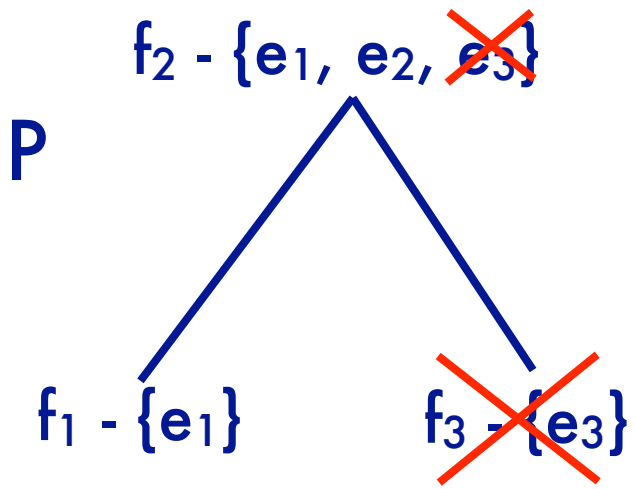
- path poset = closed sets of $E(T_f)$ ordered by inclusion

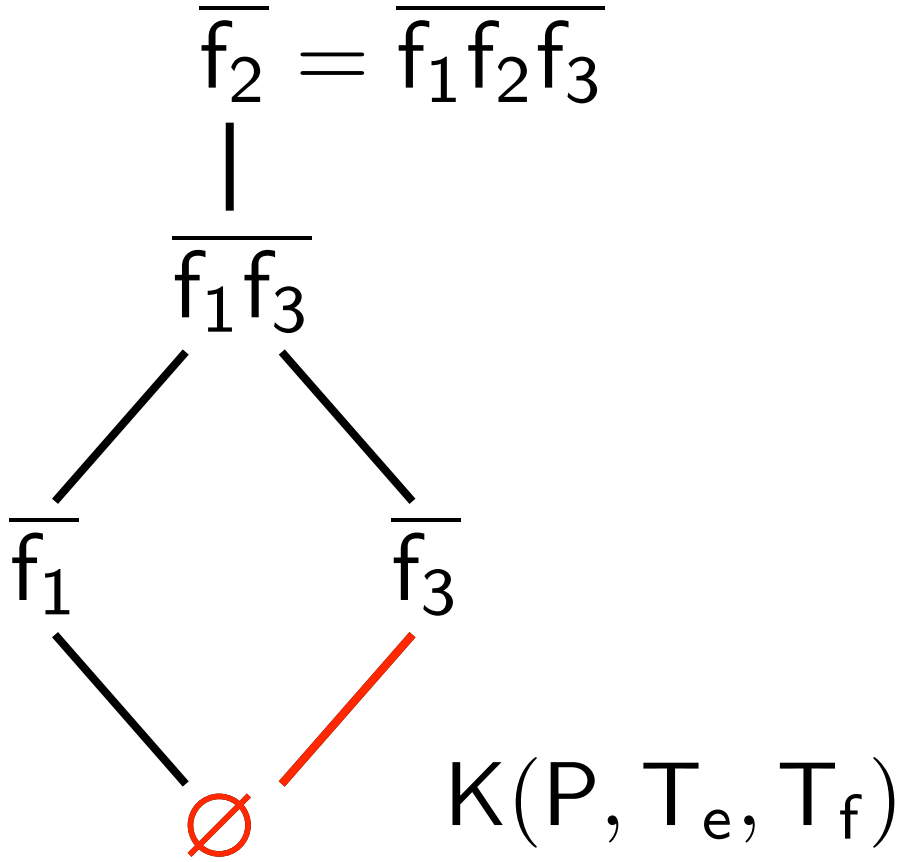




1: Path Poset - $K(P, T_e, T_f)$

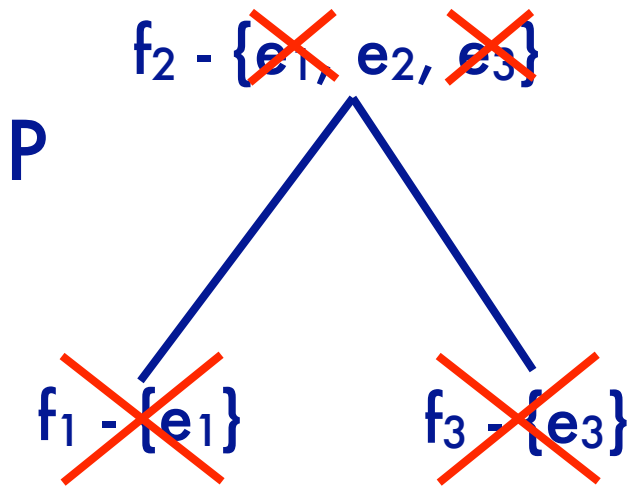
- path poset = closed sets of $E(T_f)$ ordered by inclusion

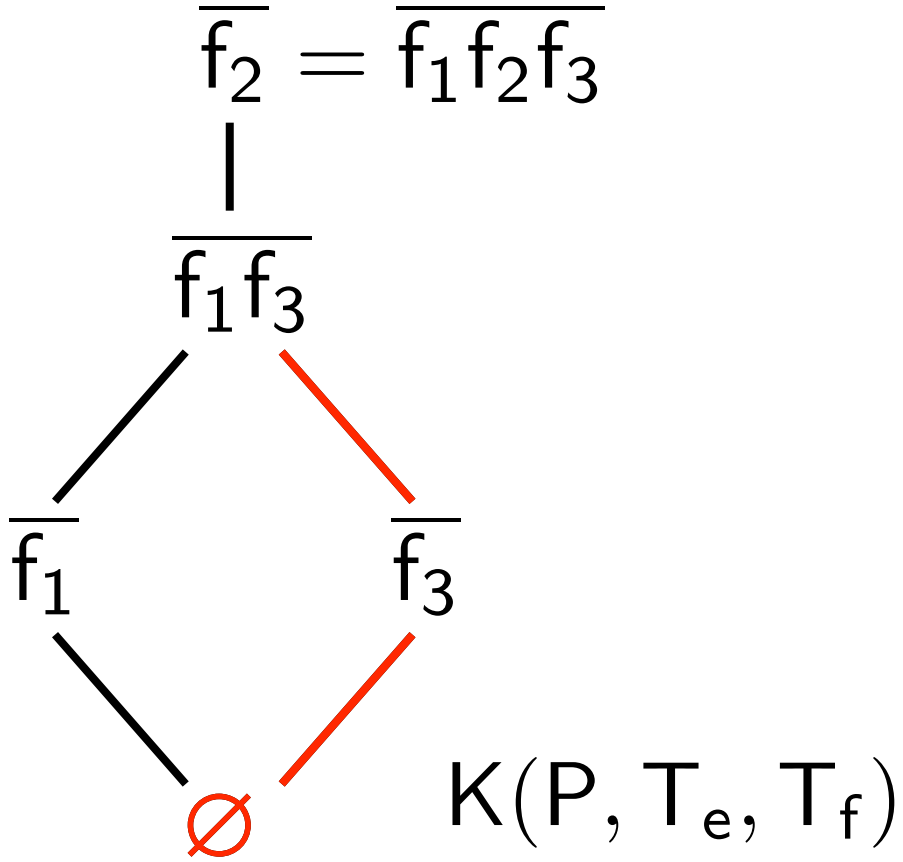




1: Path Poset - $K(P, T_e, T_f)$

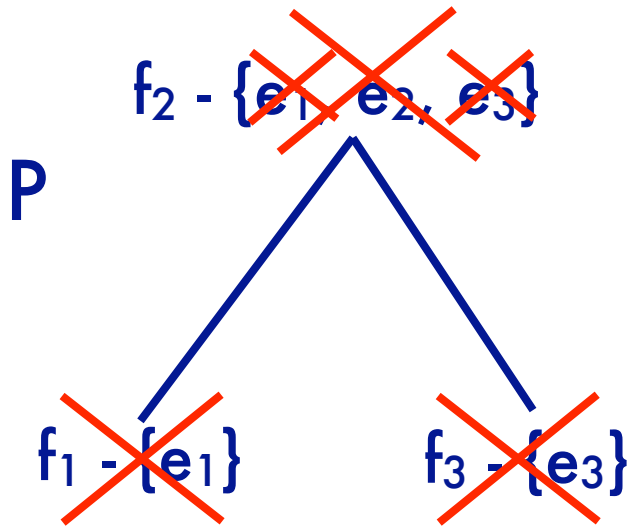
- path poset = closed sets of $E(T_f)$ ordered by inclusion

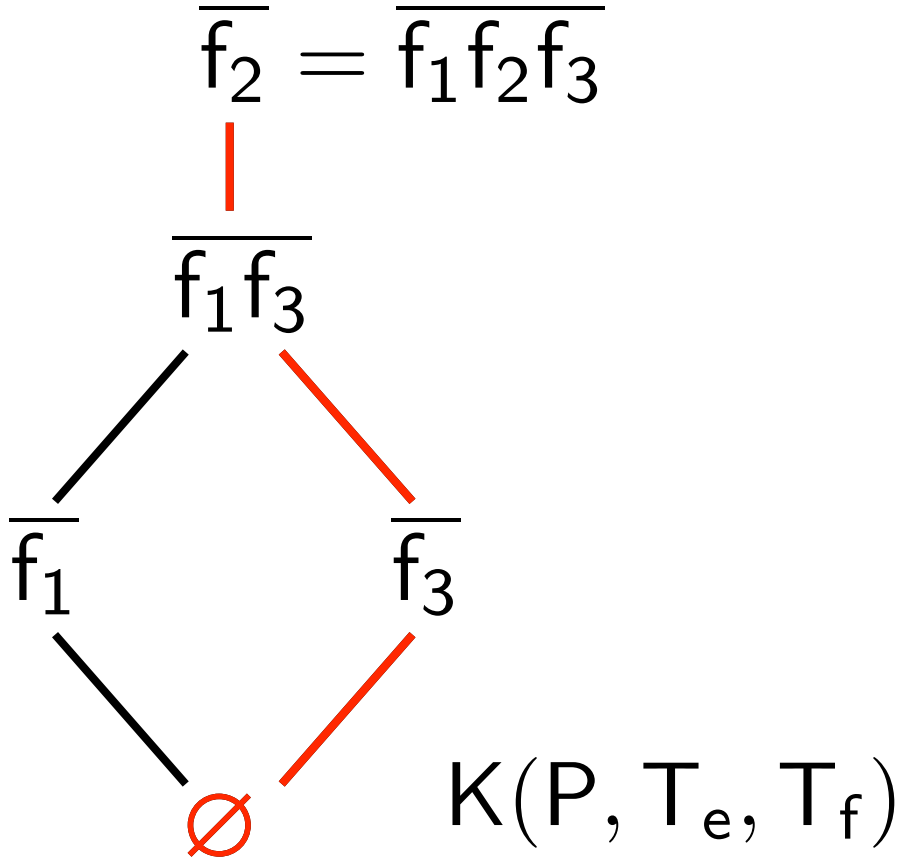




1: Path Poset - $K(P, T_e, T_f)$

- path poset = closed sets of $E(T_f)$ ordered by inclusion

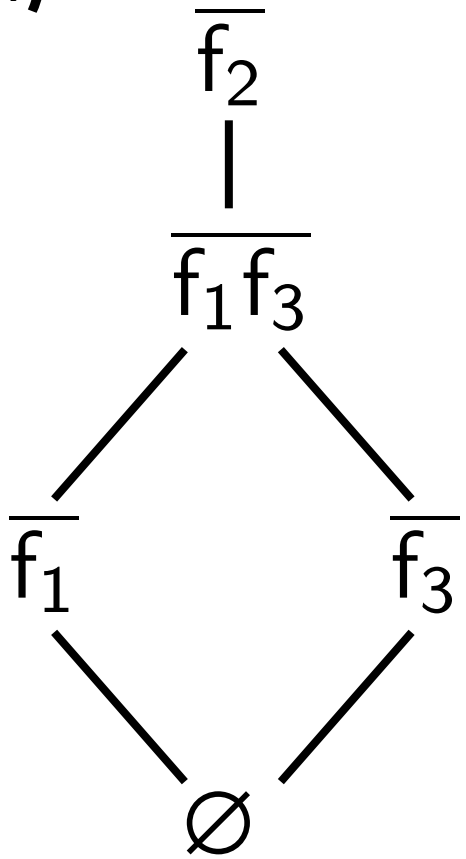




$K(P, T_e, T_f)$

1: Path Poset

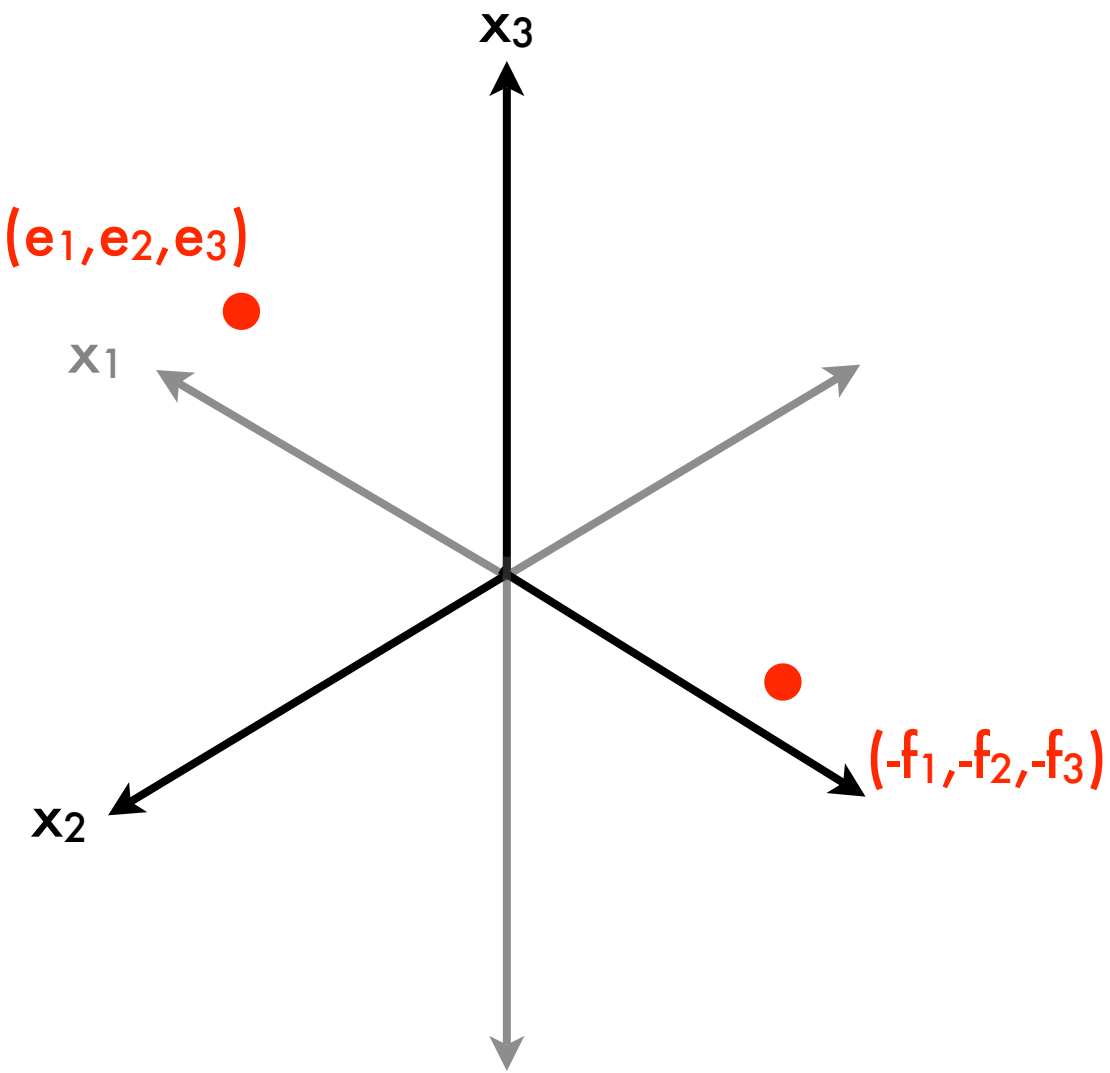
- maximal chains in $K(P, T_e, T_f)$ $\leftrightarrow$ maximal path spaces



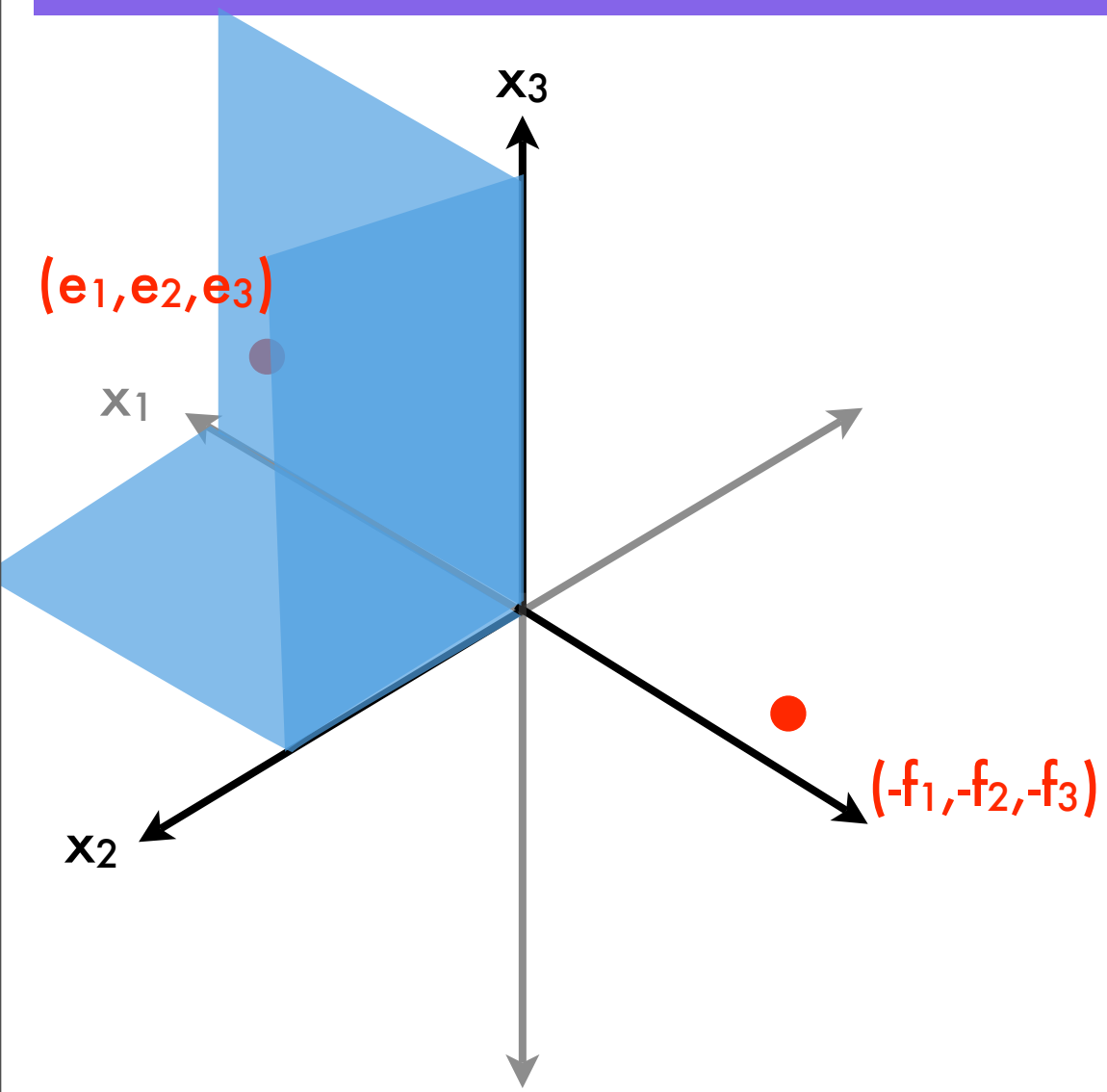
2: Local Geodesics

- given a path space, the *local geodesic* is the shortest path between the two trees going through that path space
- geodesic = min{local geodesic for each path space in S}

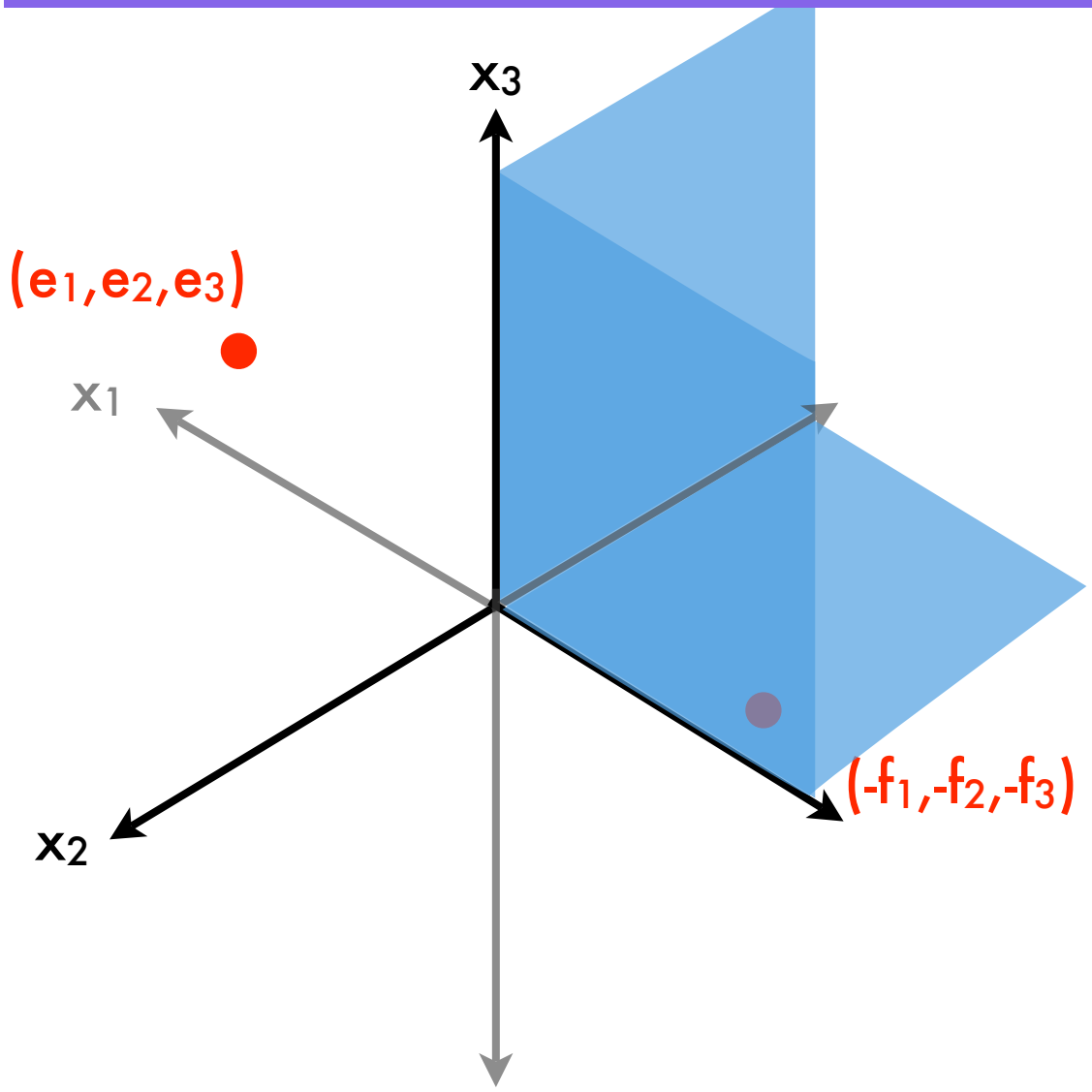
2: Isometric to part of R^k



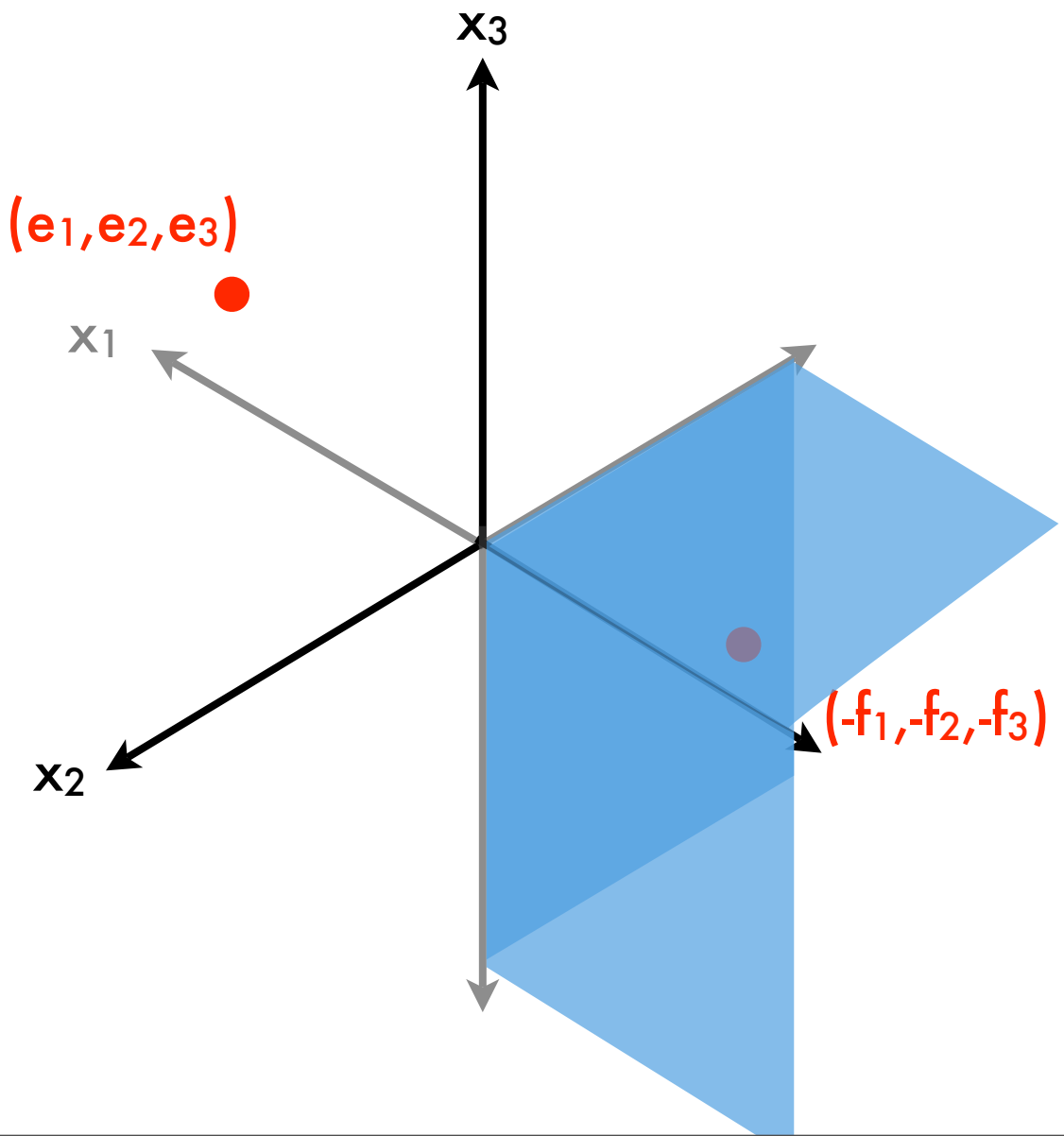
2: Isometric to part of R^k



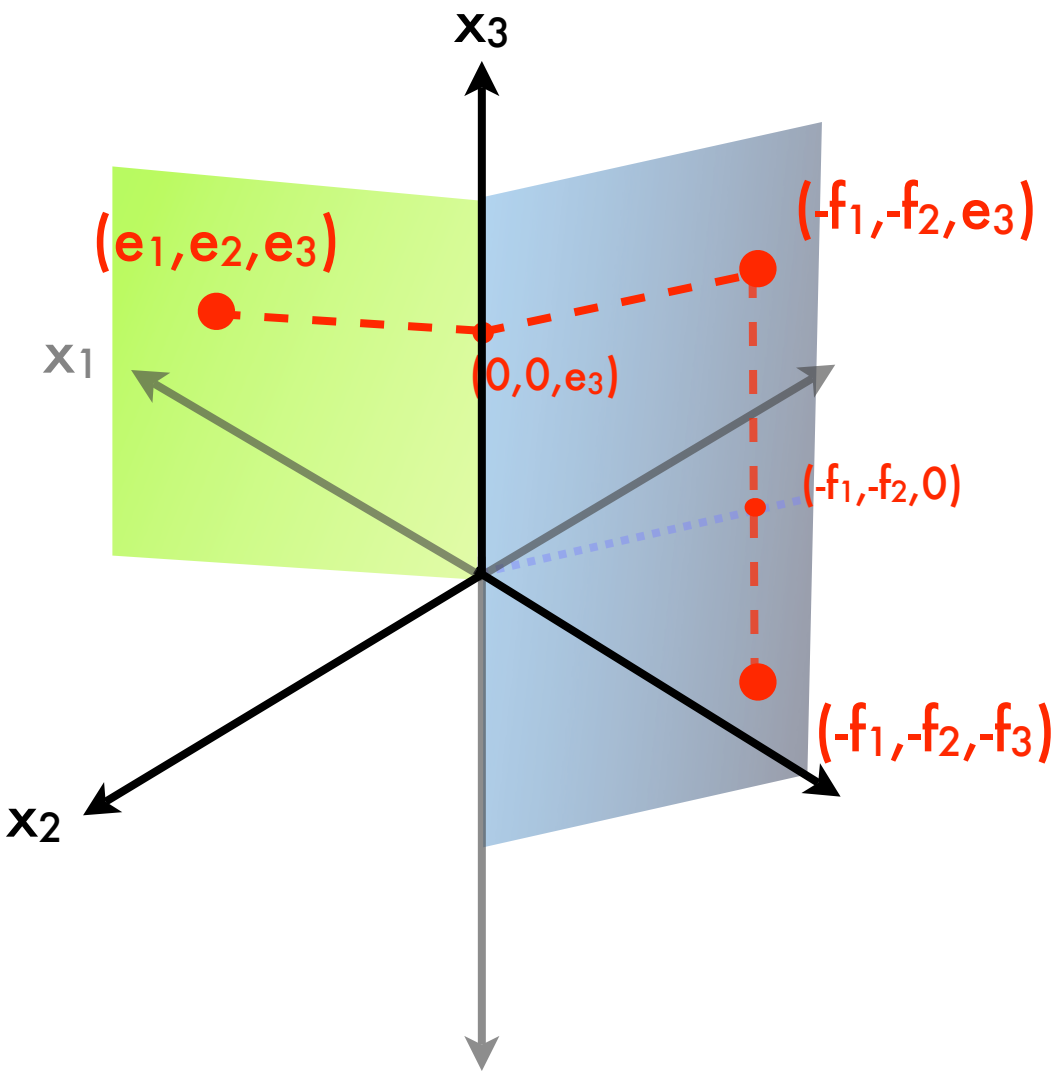
2: Isometric to part of R^k



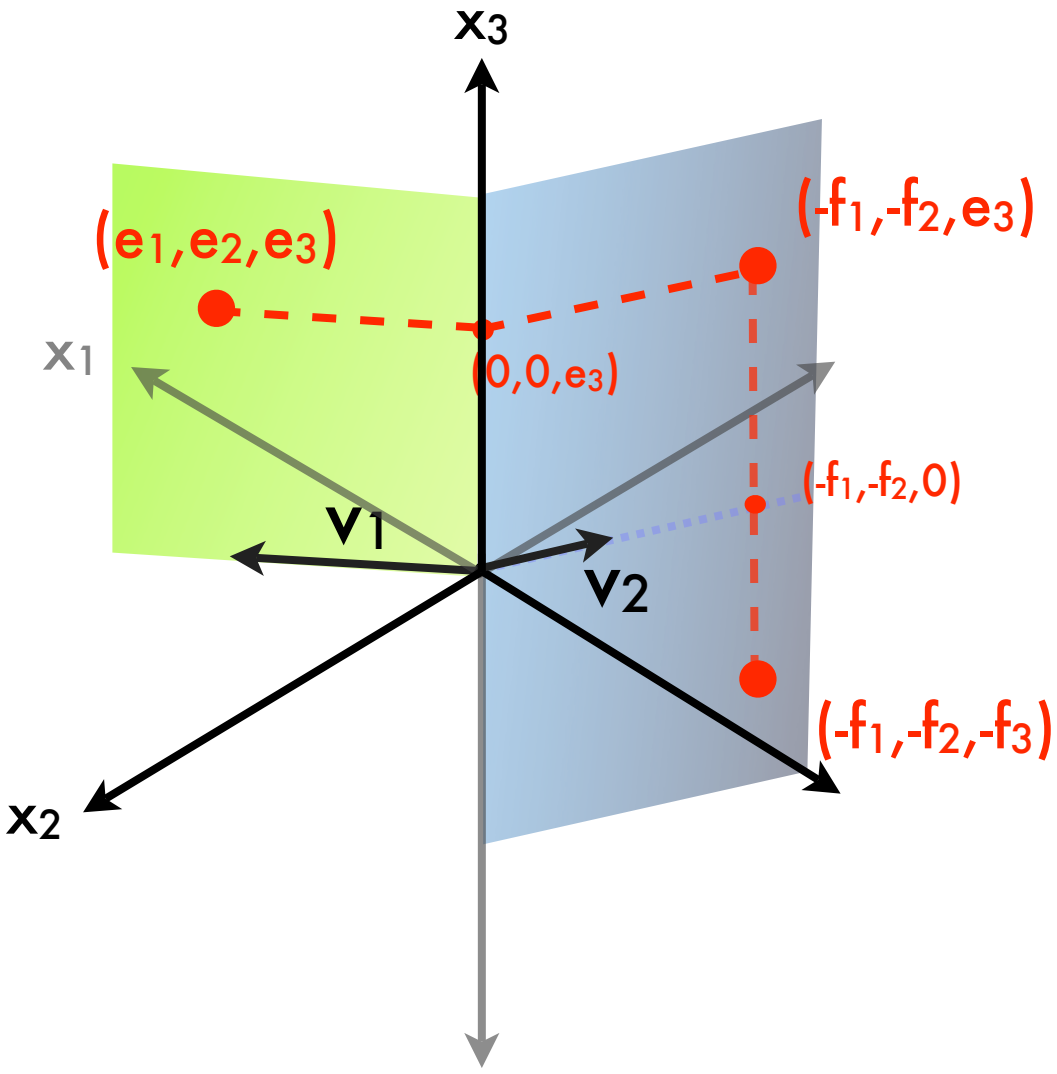
2: Isometric to part of R^k

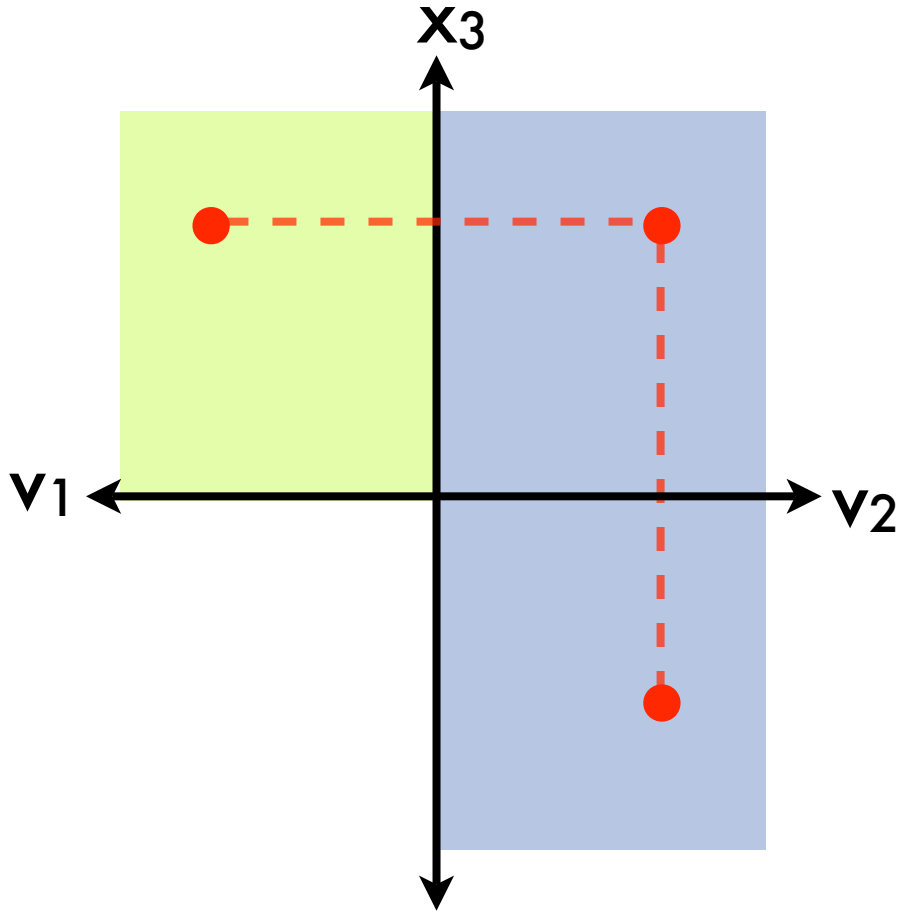


2: Isometric to part of R^k

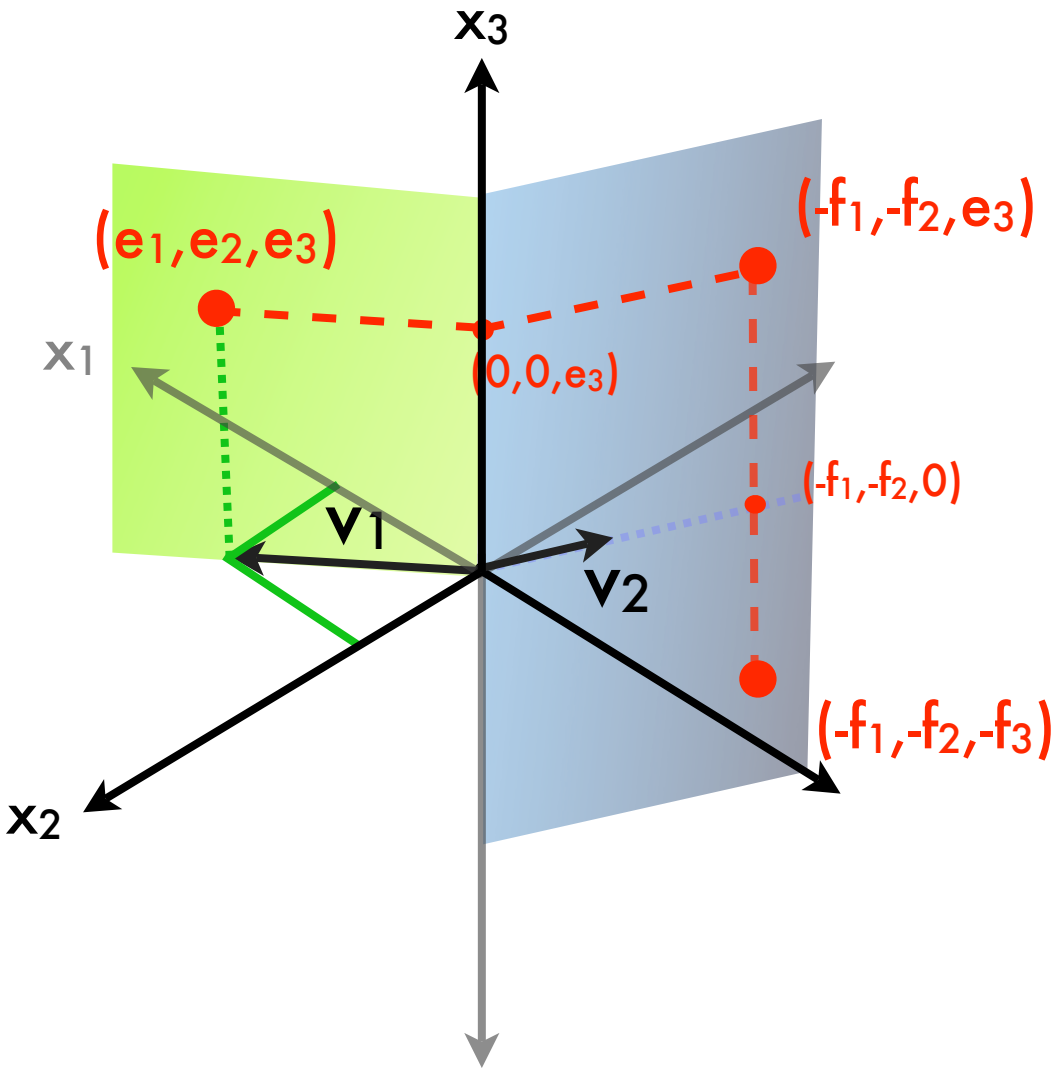


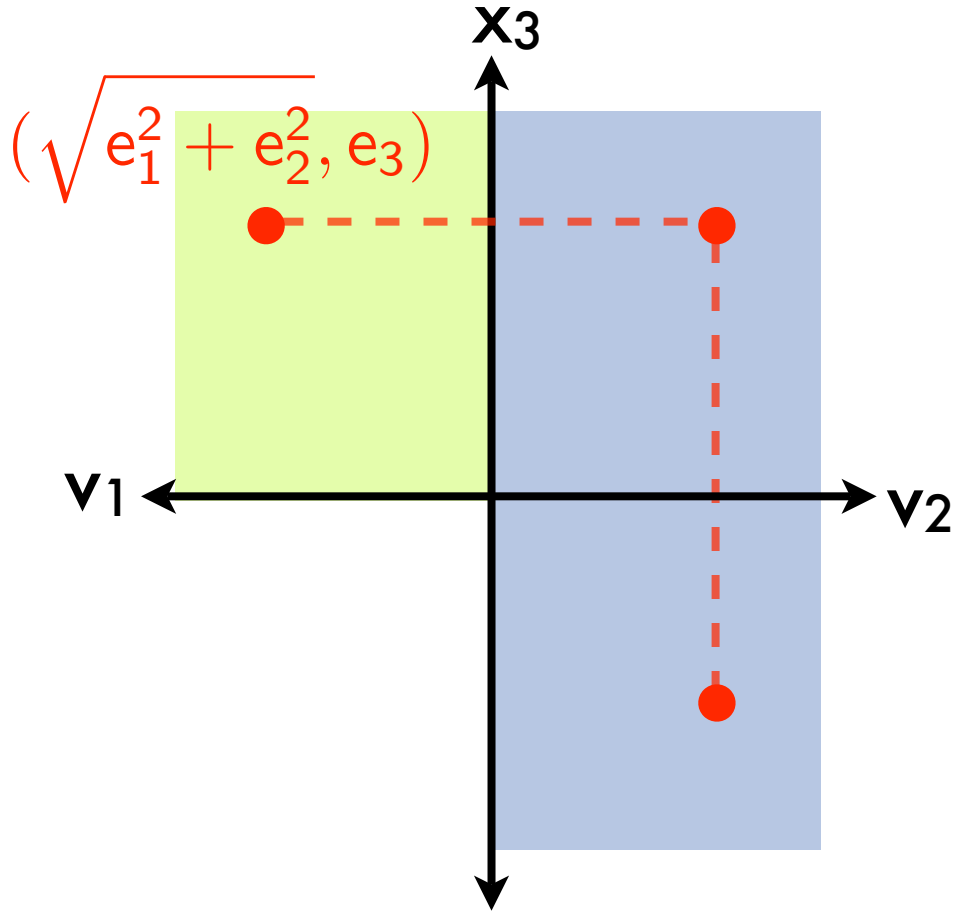
2: Isometric to part of R^k



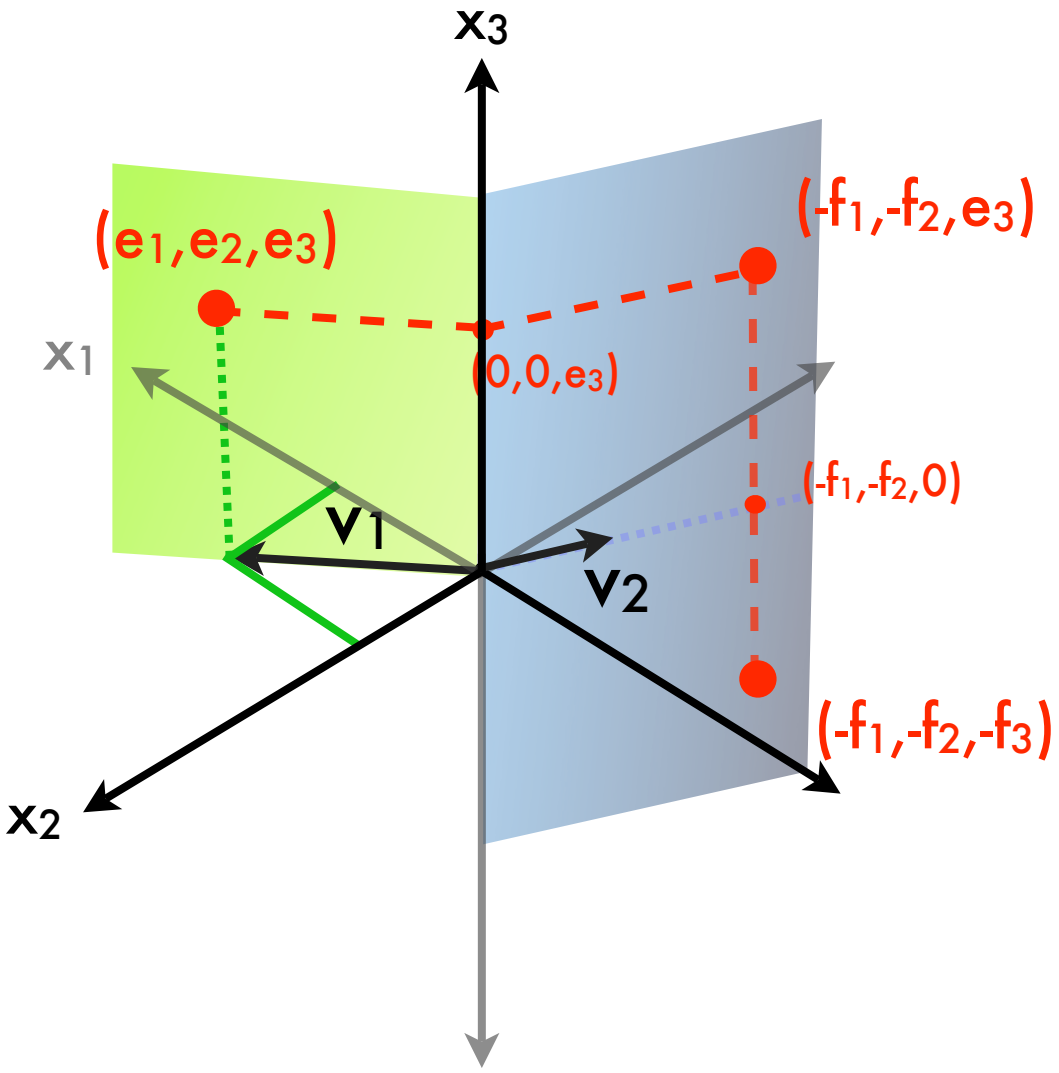


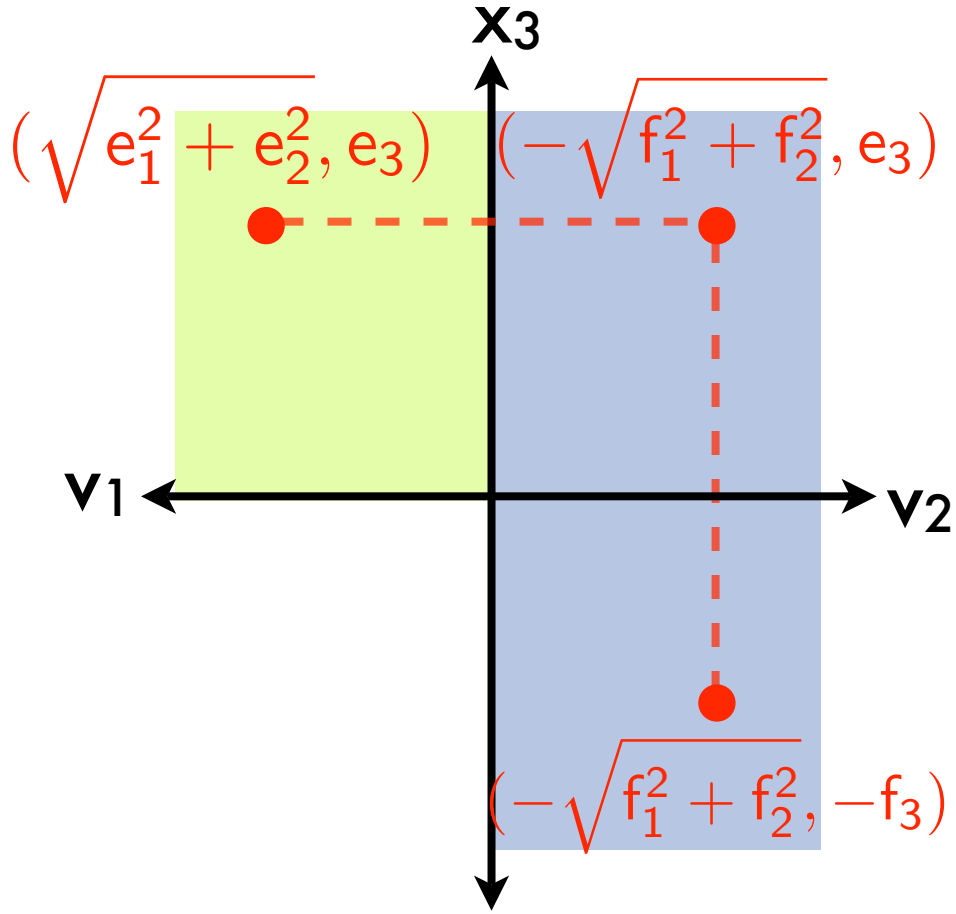
2: Isometric to part of R^k



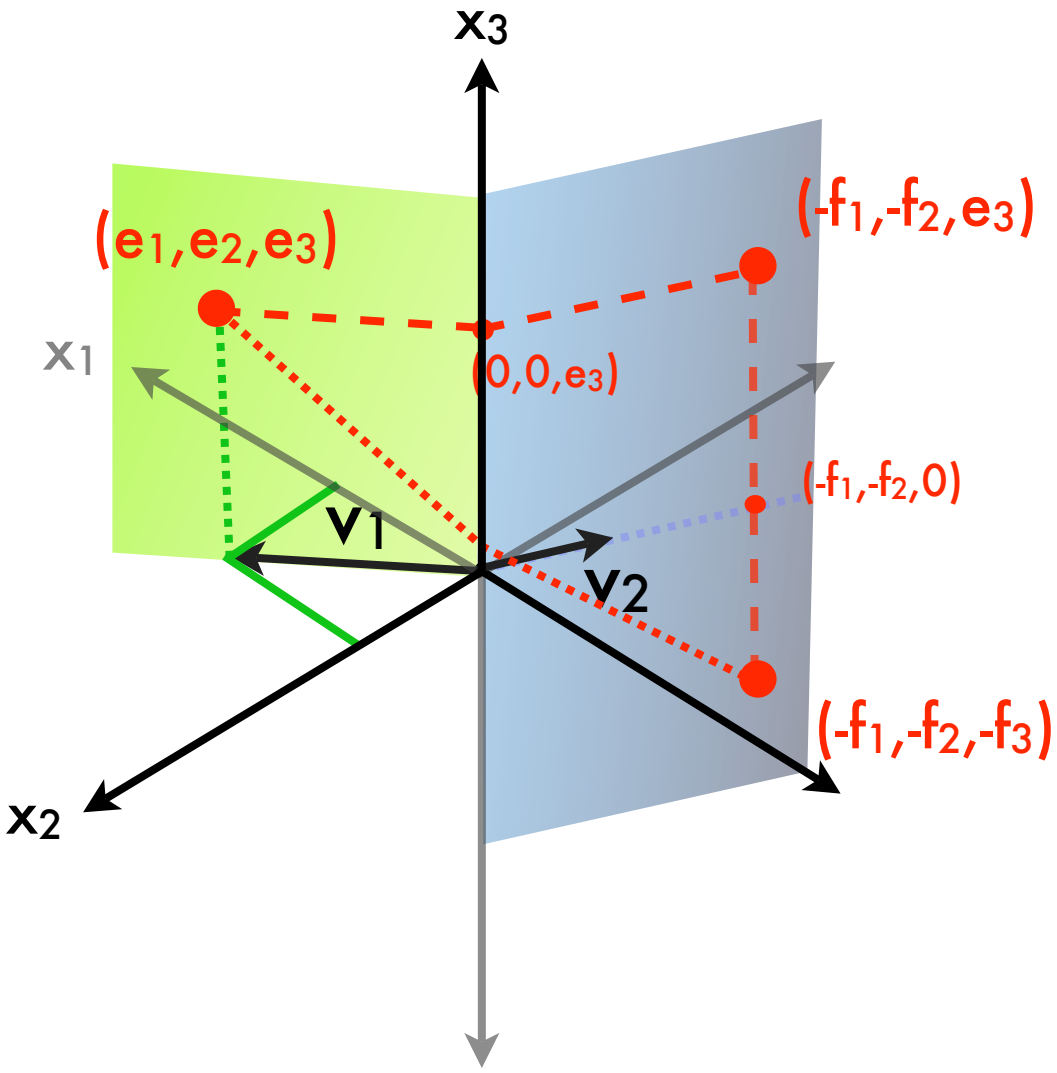


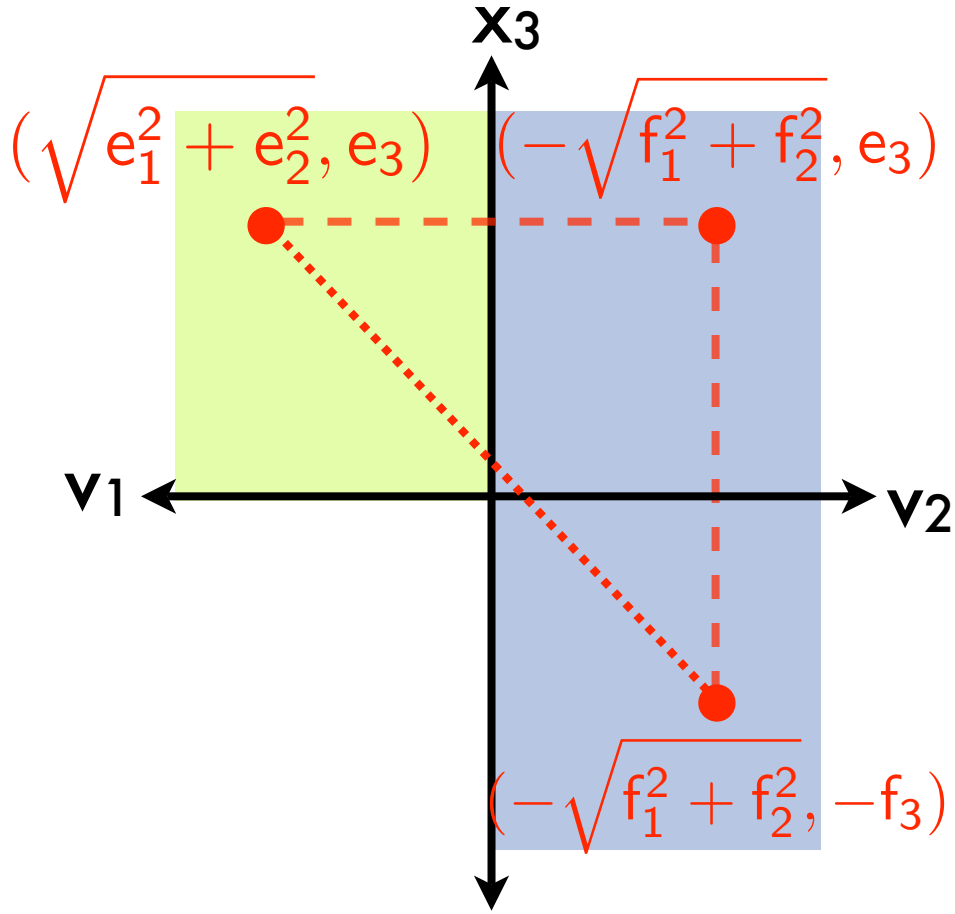
2: Isometric to part of R^k





2: Isometric to part of R^k

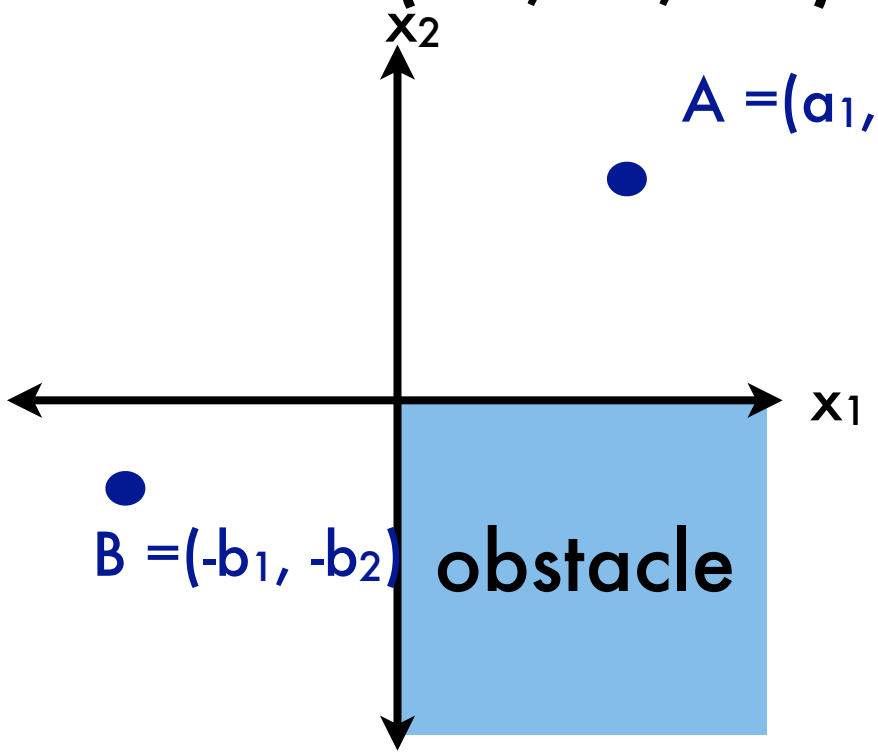




..... = geodesic

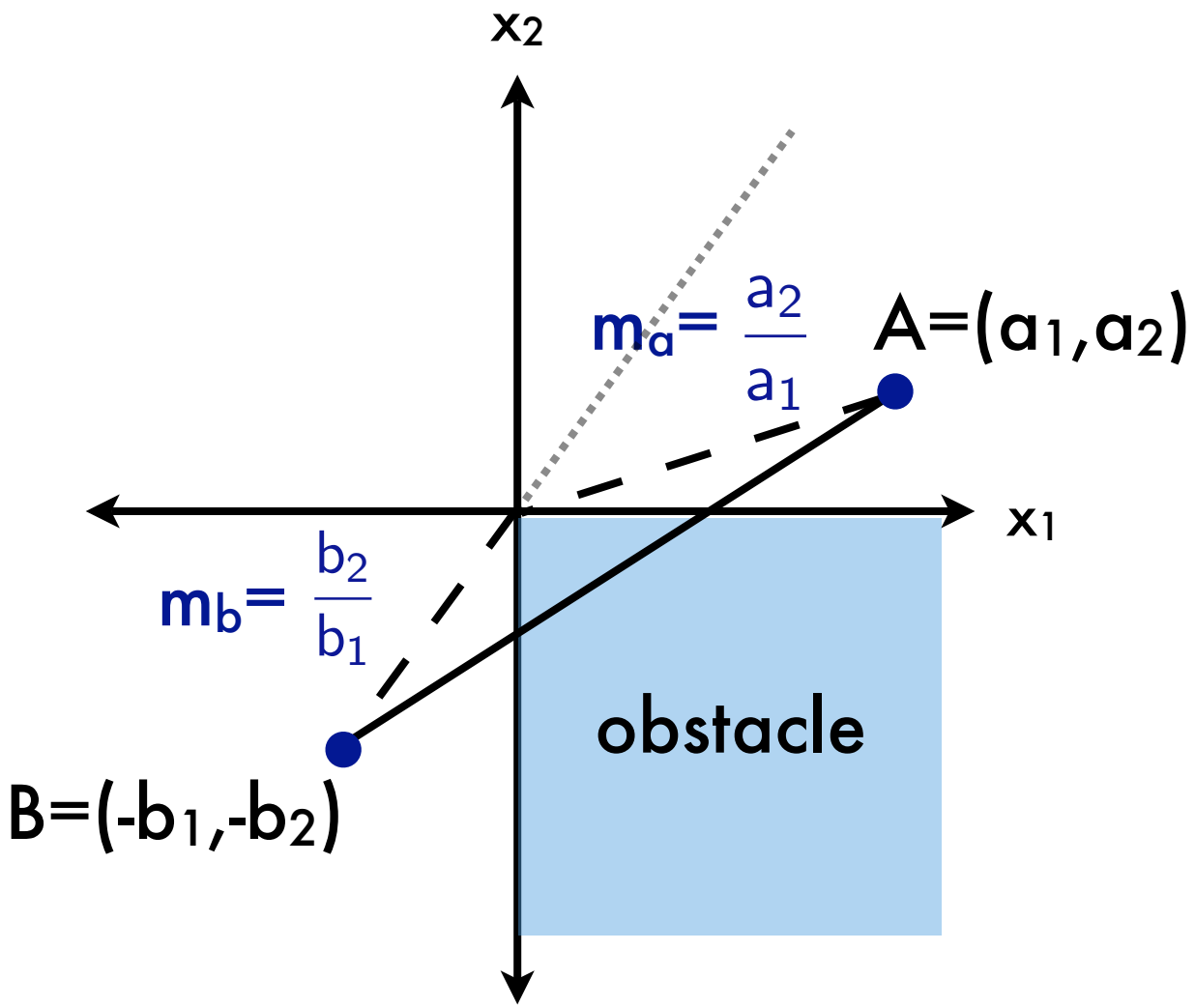
2: Reformulated Problem

- each group of edges dropped/added together corresponds to a dimension
- find Euclidean shortest path from $A = (a_1, \ldots, a_k)$ to $B = (-b_1, \ldots, -b_k)$ where a_i, b_i positive for all i



i^{th} orthant $= (\underset{i}{\leq 0, \ldots, \leq 0}, \geq 0, \ldots, \geq 0)$

2: Geodesic Conditions



$\overline{AB}$ is the shortest path

$\Updownarrow$

$m_b \leq m_a$

or $\frac{a_1}{b_1} \leq \frac{a_2}{b_2}$

- in higher dimensions, project onto the $x_i x_j$-plane

2: Ratio Condition

- if $\frac{a_1}{b_1} \leq \frac{a_2}{b_2} \leq \ldots \leq \frac{a_k}{b_k}$, then $\overline{AB}$ is

  the shortest obstacle avoiding path

  $\Rightarrow$ distance = Euclidean distance

$$= \sqrt{\sum_{i=1}^{k} (a_i + b_i)^2}$$

2: Key Lemma

- if $\frac{a_1}{b_1} \leq \frac{a_2}{b_2} \leq \ldots \leq \frac{a_i}{b_i} > \frac{a_{i+1}}{b_{i+1}}$, then any shortest path goes through the intersection of the $(i-1)^{th}$, i^{th}, and $(i+1)^{th}$ orthants

  $\Rightarrow$ we should replace the ratios $\frac{a_i}{b_i}$ and $\frac{a_{i+1}}{b_{i+1}}$ with the ratio $\frac{\sqrt{a_i^2 + a_{i+1}^2}}{\sqrt{b_i^2 + b_{i+1}^2}}$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

ok

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

ok

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

apply lemma

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \cancel{\frac{3}{1} > \frac{2}{3}} \leq \frac{5}{1}$$

$$\frac{\sqrt{3^2 + 2^2}}{\sqrt{1^2 + 3^2}}$$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

ok

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

apply lemma

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

$$\frac{\sqrt{2^2 + \sqrt{13}^2}}{\sqrt{1^2 + \sqrt{10}^2}}$$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

$$\frac{1}{1} \leq \frac{\sqrt{17}}{\sqrt{11}} \leq \frac{5}{1}$$

2: R^k Algorithm

- To find the shortest path from A = (1, 2, 3, 2, 5) to B = (-1, -1, -1, -3, -1) contained in the selected orthants:

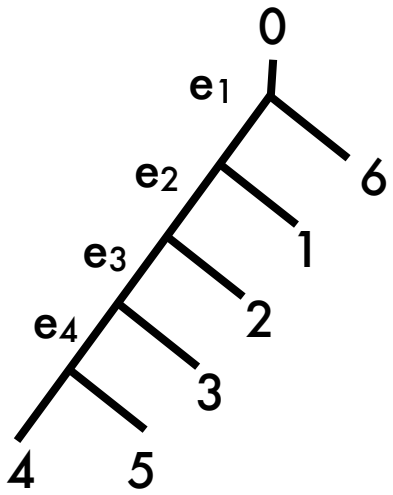
$$\frac{1}{1} \leq \frac{2}{1} \leq \frac{3}{1} > \frac{2}{3} \leq \frac{5}{1}$$

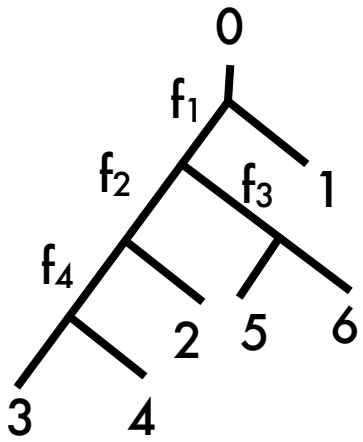
$$\frac{1}{1} \leq \frac{2}{1} > \frac{\sqrt{13}}{\sqrt{10}} \leq \frac{5}{1}$$

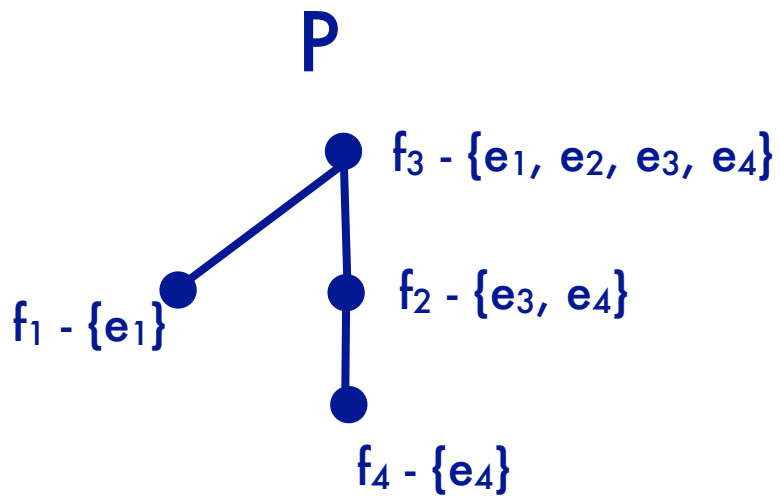
$$\frac{1}{1} \leq \frac{\sqrt{17}}{\sqrt{11}} \leq \frac{5}{1}$$

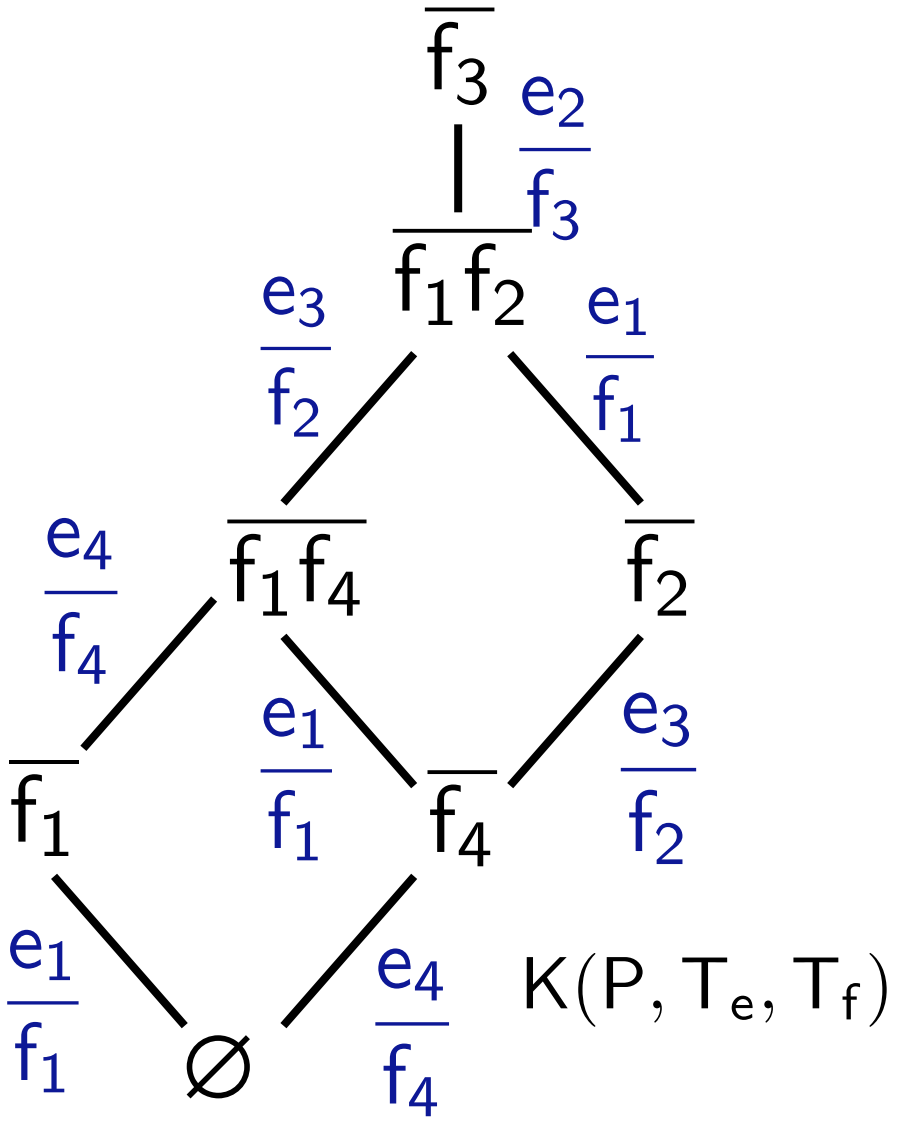
dist = $||(1, \sqrt{17}, 5) - (-1, -\sqrt{11}, -1)|| = 11.0448...$

3: Putting It All Together

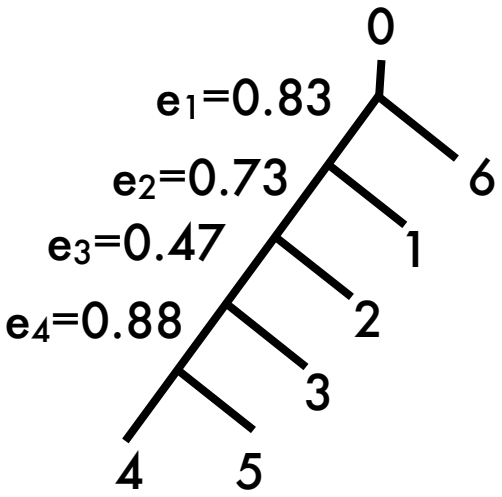


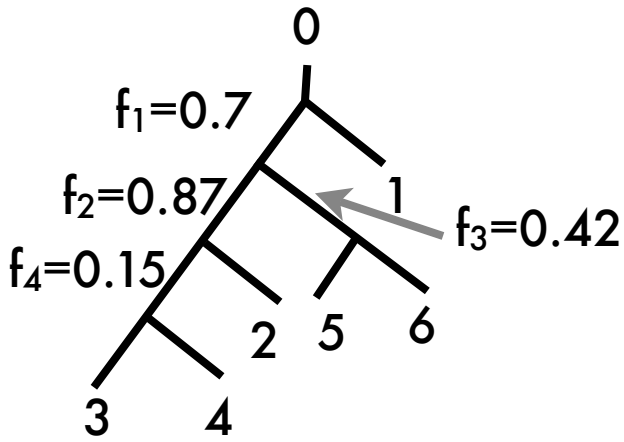






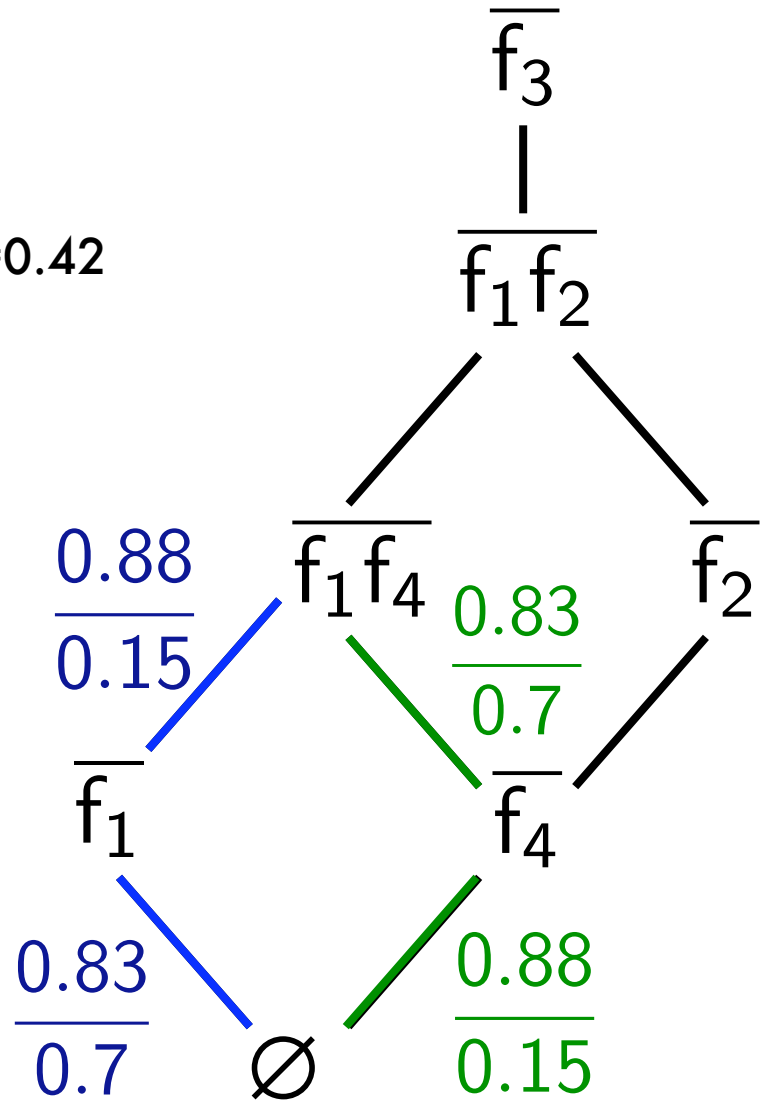
3: Putting It All Together



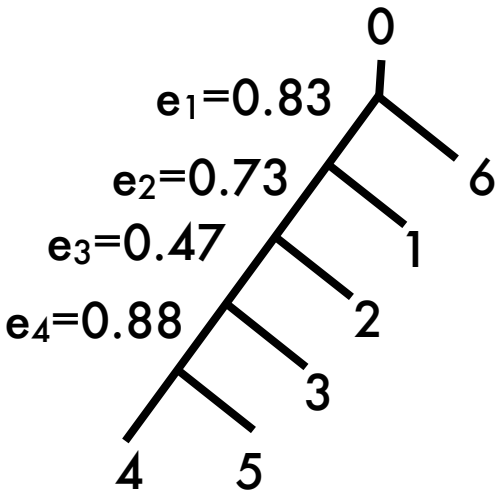


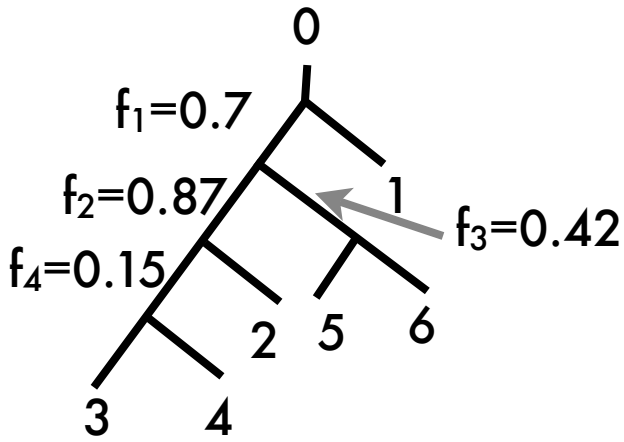
$$d_E\left(\frac{0.83}{0.7}, \frac{0.88}{0.15}\right) = 1.84$$

$$d_E\left(\frac{0.88}{0.15}, \frac{0.83}{0.7}\right) = 1.95$$



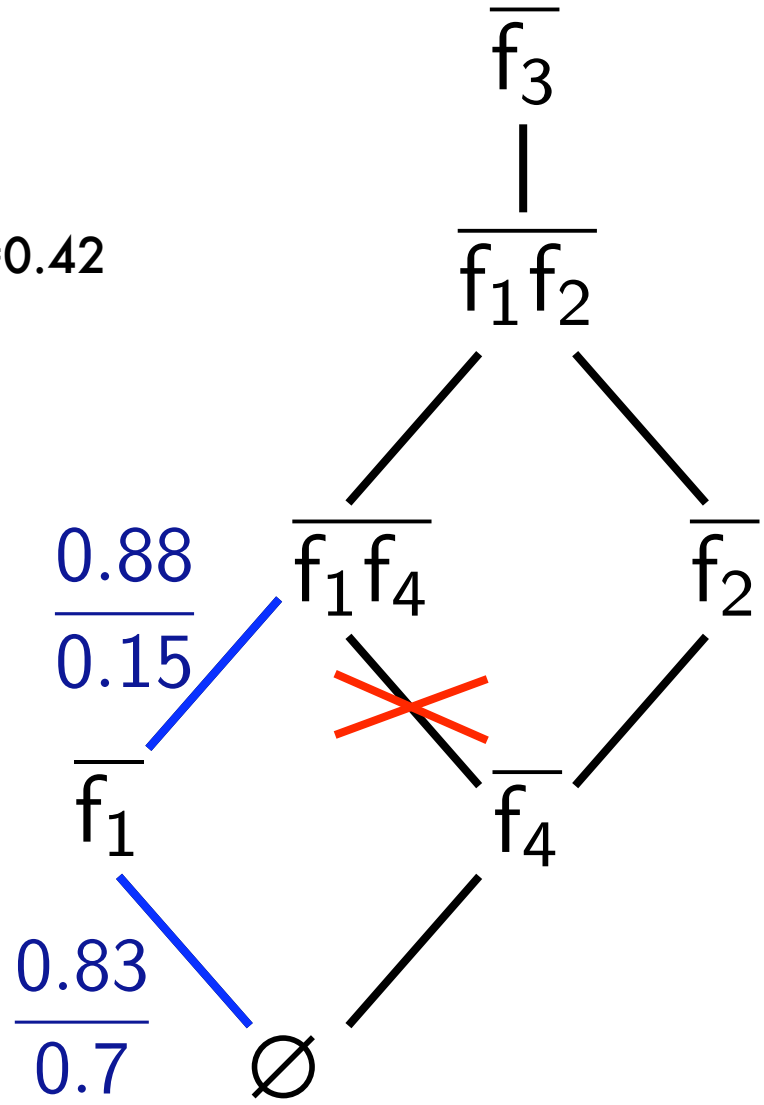
3: Putting It All Together



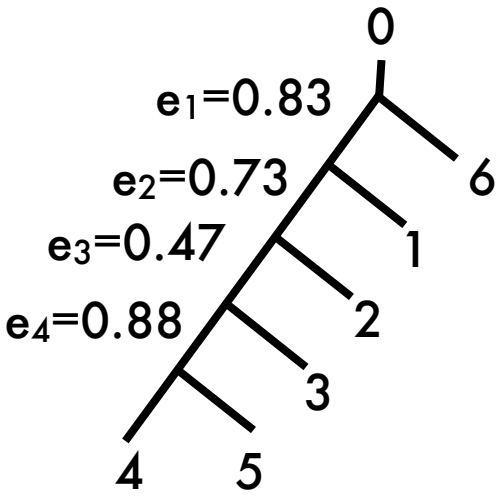


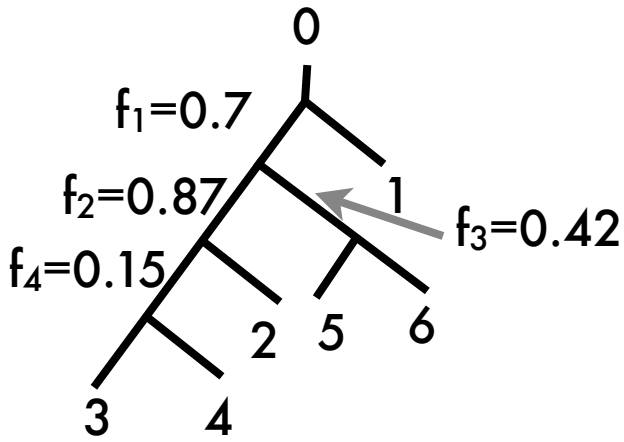
$$d_E\left(\frac{0.83}{0.7}, \frac{0.88}{0.15}\right) = 1.84$$

$$d_E\left(\frac{0.88}{0.15}, \frac{0.83}{0.7}\right) = 1.95$$



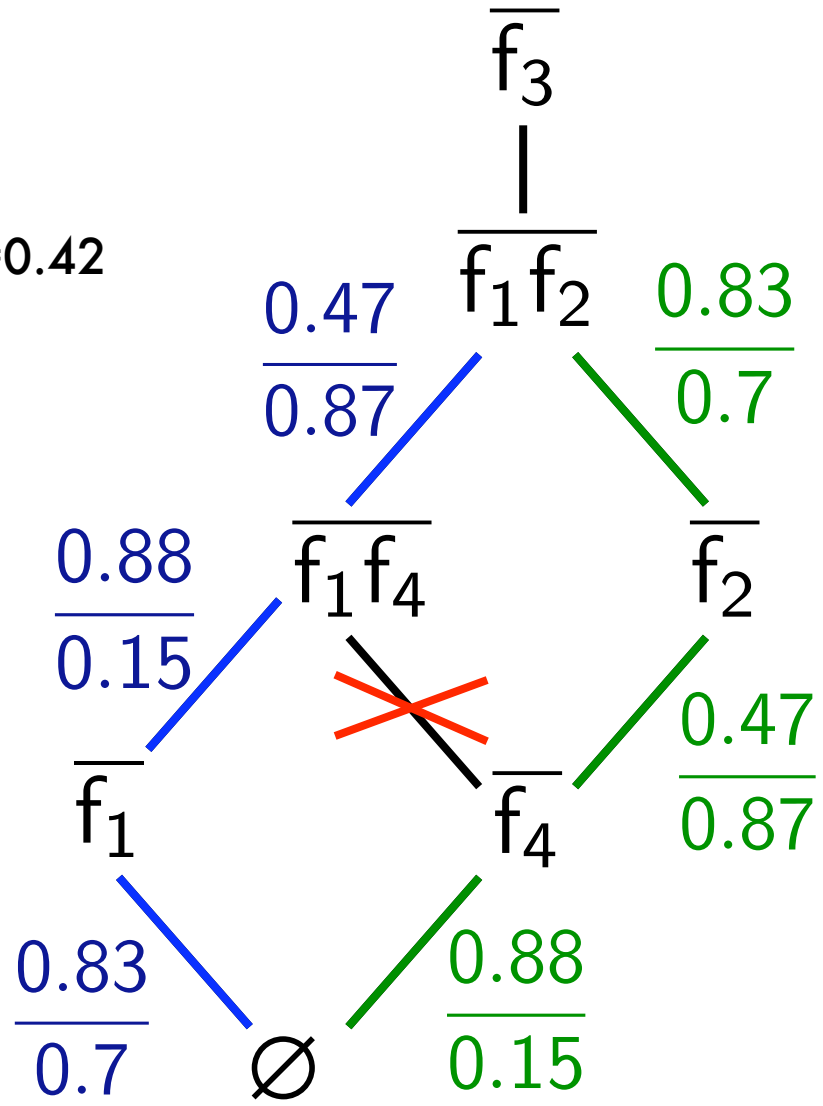
3: Putting It All Together



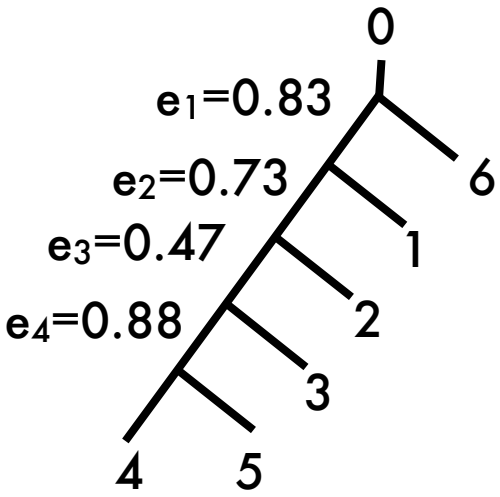


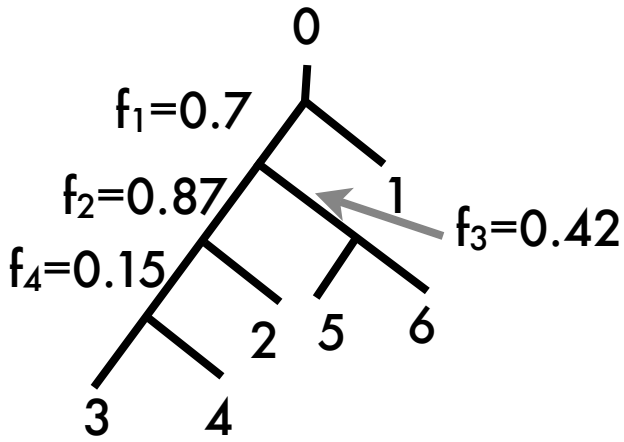
$$d_E\left(\frac{0.83}{0.7}, \frac{0.88}{0.15}, \frac{0.47}{0.87}\right) = 2.4244$$

$$d_E\left(\frac{0.88}{0.15}, \frac{0.47}{0.87}, \frac{0.83}{0.7}\right) = 2.4243$$



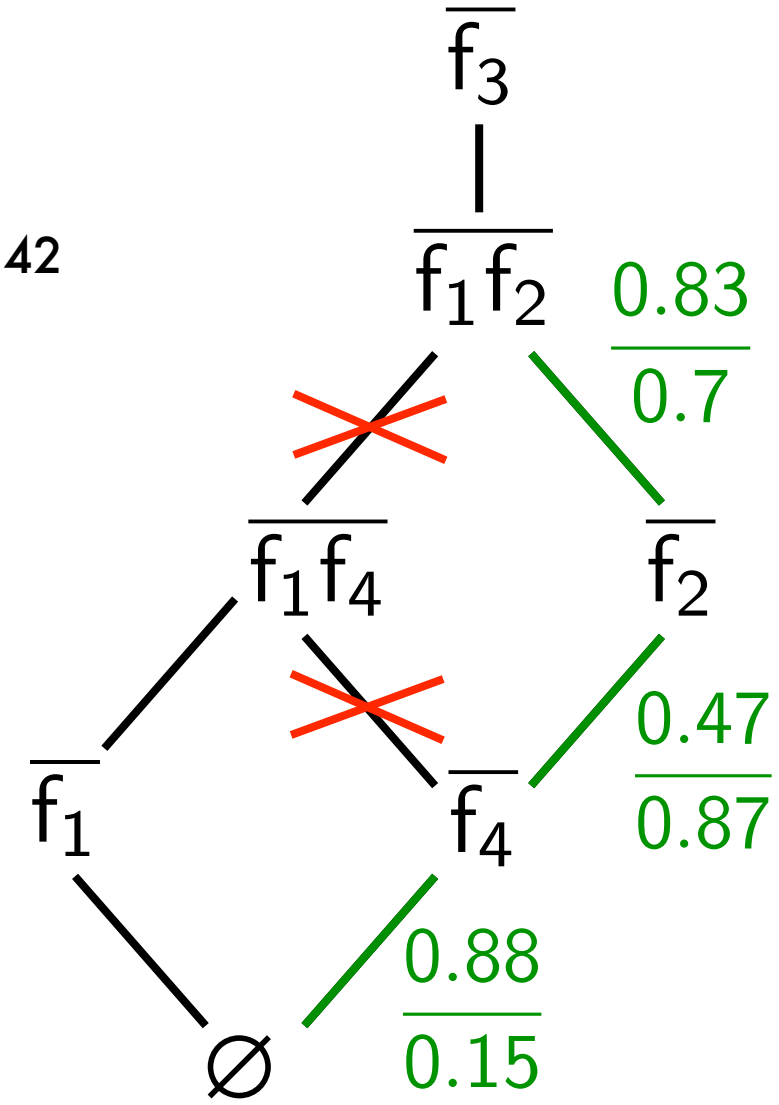
3: Putting It All Together



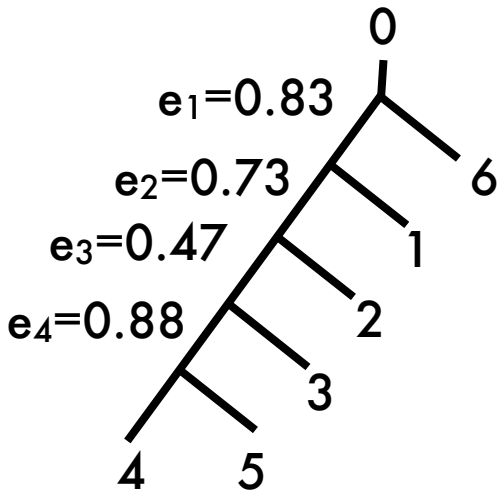


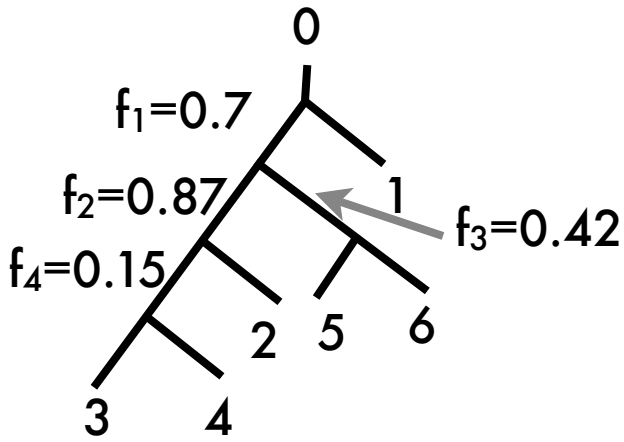
$$d_E\left(\frac{0.83}{0.7}, \frac{0.88}{0.15}, \frac{0.47}{0.87}\right) = 2.4244$$

$$d_E\left(\frac{0.88}{0.15}, \frac{0.47}{0.87}, \frac{0.83}{0.7}\right) = 2.4243$$



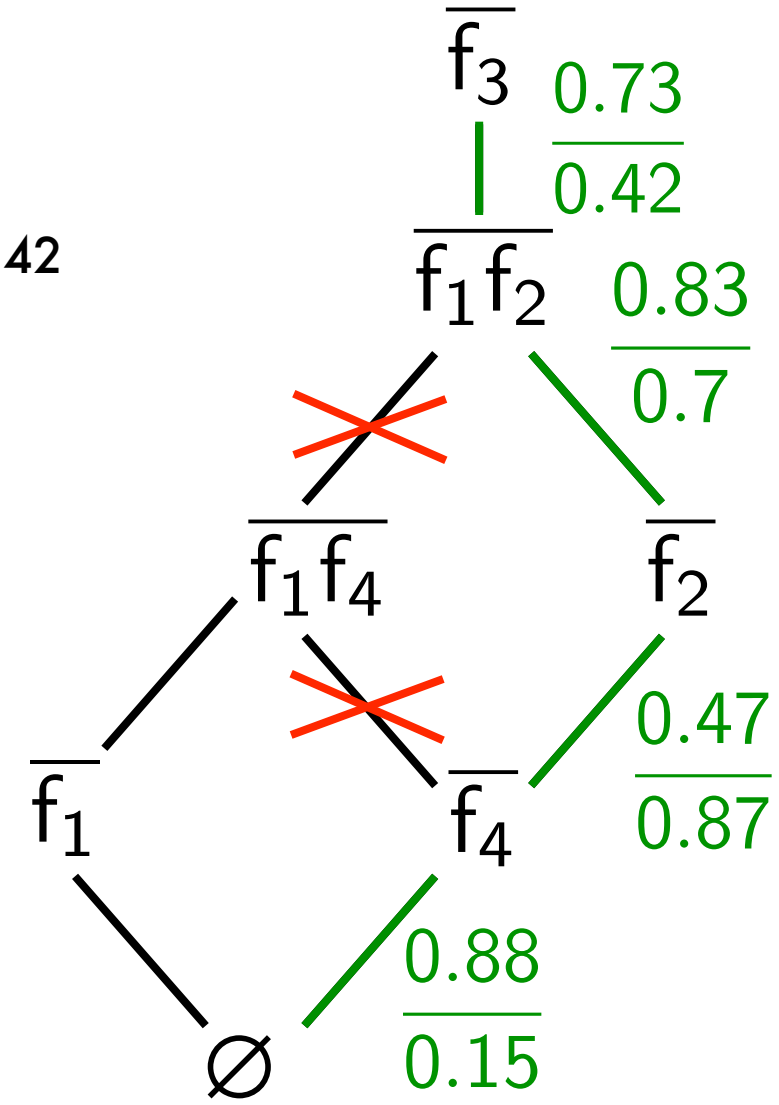
3: Putting It All Together





- geodesic distance is

$$d_E\left(\frac{0.88}{0.15}, \frac{0.47}{0.87}, \frac{0.83}{0.7}, \frac{0.73}{0.42}\right) = 2.65$$



Complexity

- worst case: # of nodes in path poset is exponential in n (number of leaves)
- averaged 2.2 sec. to calculate geodesic distance between 43-taxa trees of archaea and bacteria

Thank You

- acknowledgements:
  - Lou Billera
  - Karen Vogtmann
  - Joe Mitchell
  - Philippe Lopez (data set)
  - NSF

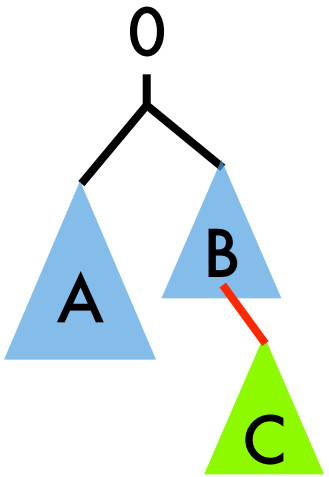
Shared Edges

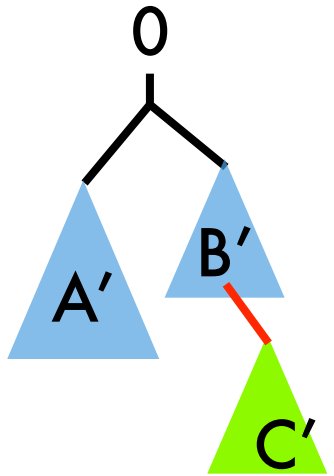
1. decompose tree along shared edges
2. apply algorithm to each subtree
3. combine to get geodesic

Shared Edges

- Billera, Holmes, and Vogtmann tell us:
  If e is a shared edge, every tree on the geodesic also contains e

1. decompose tree along shared edges





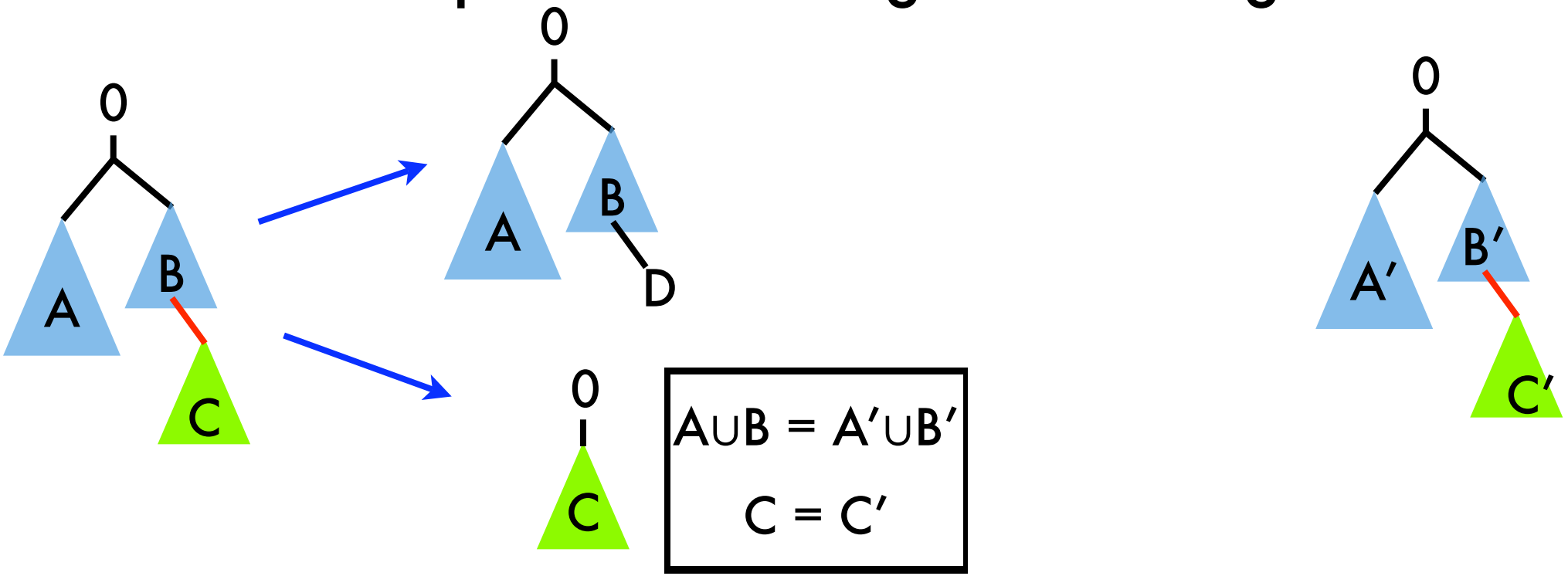
$$A \cup B = A' \cup B'$$

$$C = C'$$

Shared Edges

- Billera, Holmes, and Vogtmann tell us:
  If e is a shared edge, every tree on the geodesic also contains e

1. decompose tree along shared edges



Shared Edges

- Billera, Holmes, and Vogtmann tell us:
  If e is a shared edge, every tree on the geodesic also contains e

1. decompose tree along shared edges