Reconstructing Repeat-Annotated Phylogenetic Trees

Firas Swidan Michal Ziv-Ukelson Ron Y. Pinter

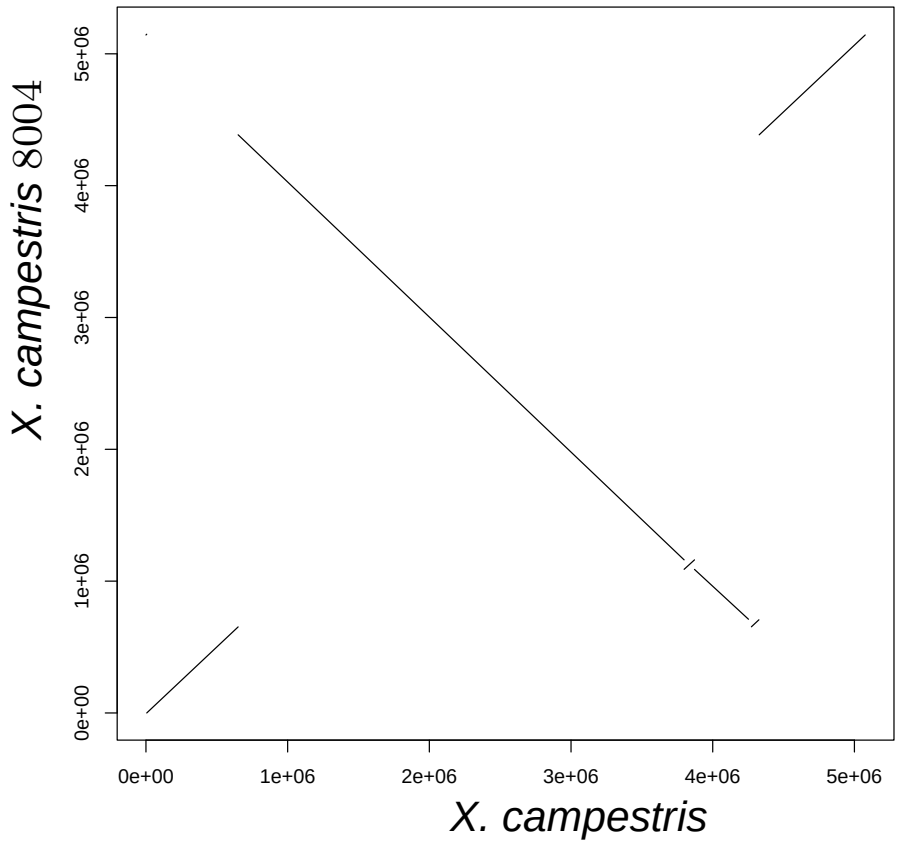
Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

Introduction

Recontstructing ancestral genome order

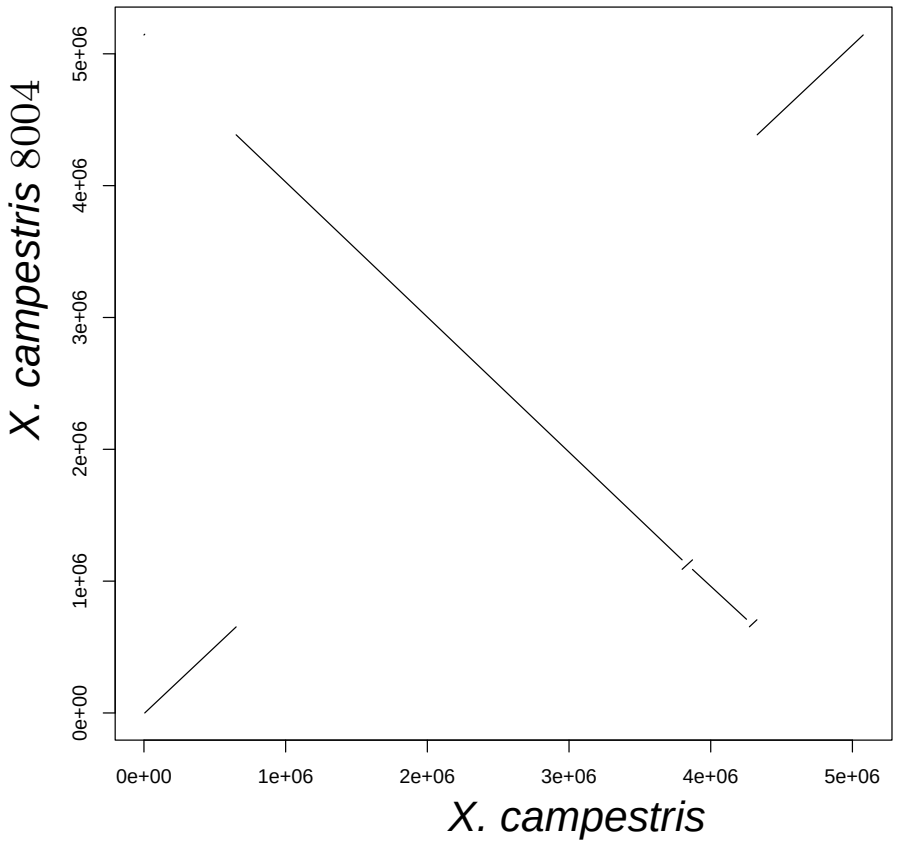
Comparing *Xanthomonas campestris* pathovar *campestris*, ATCC 33913, and 8004.

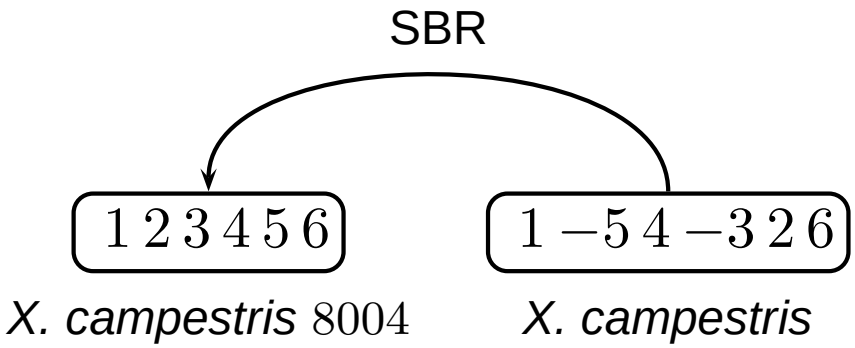


MAGIC's result [Swidan et al., 2005]

Recontstructing ancestral genome order

Comparing *Xanthomonas campestris* pathovar *campestris*, ATCC 33913, and 8004.

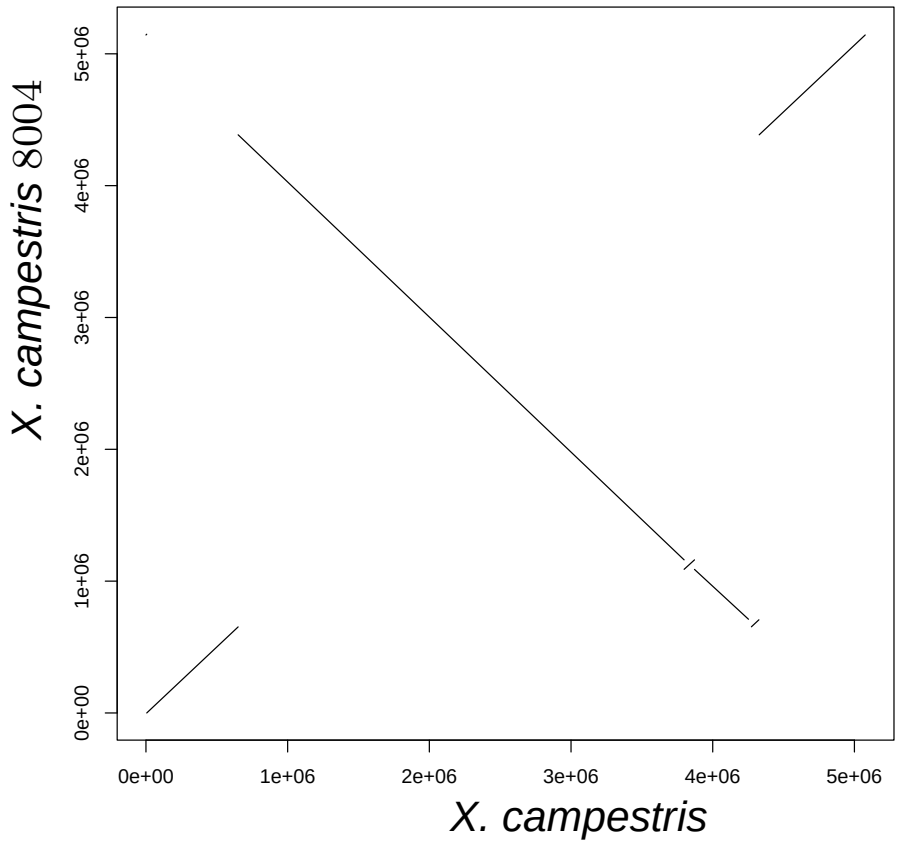


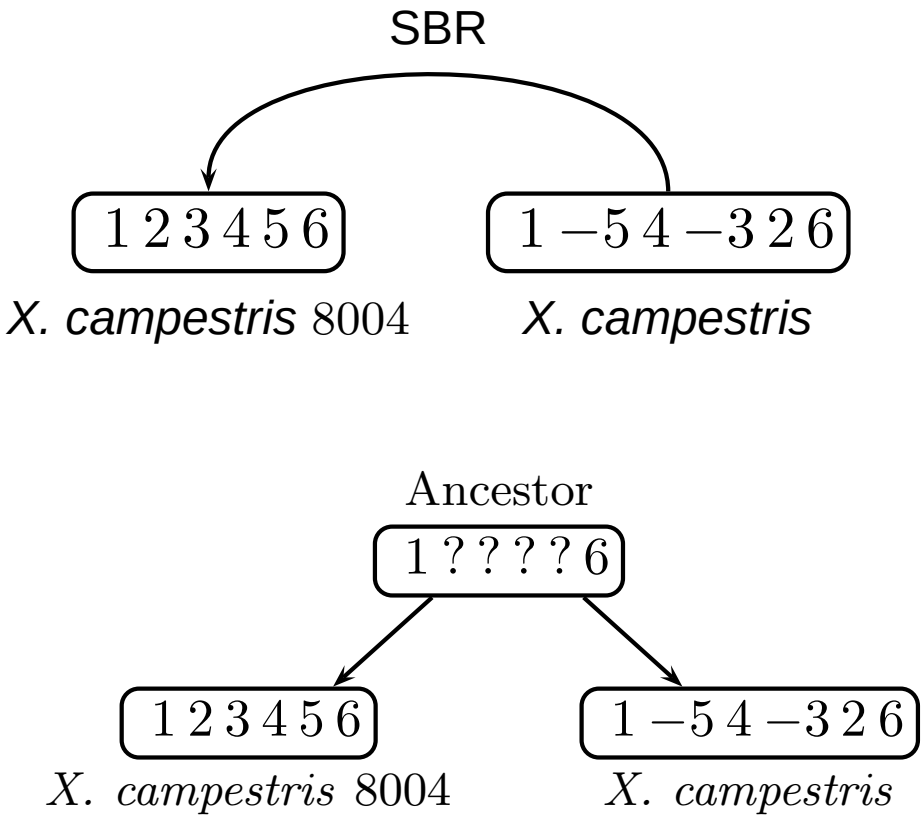


MAGIC's result [Swidan et al., 2005]

Recontstructing ancestral genome order

Comparing *Xanthomonas campestris* pathovar *campestris*, ATCC 33913, and 8004.





MAGIC's result [Swidan et al., 2005]

Mechanism

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

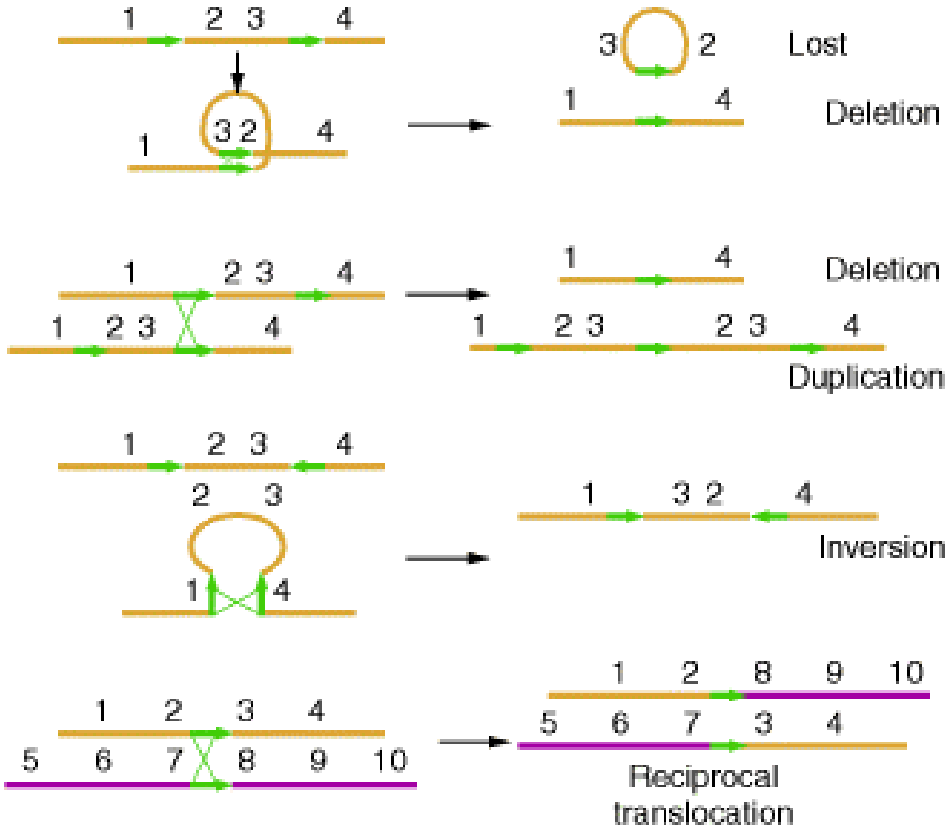
Recombinations (homologous and illegitimate) are induced by repeats [Kowalczykowski et al., 1994, Smith, 1989].

Mechanism

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

Recombinations (homologous and illegitimate) are induced by repeats [Kowalczykowski et al., 1994, Smith, 1989].

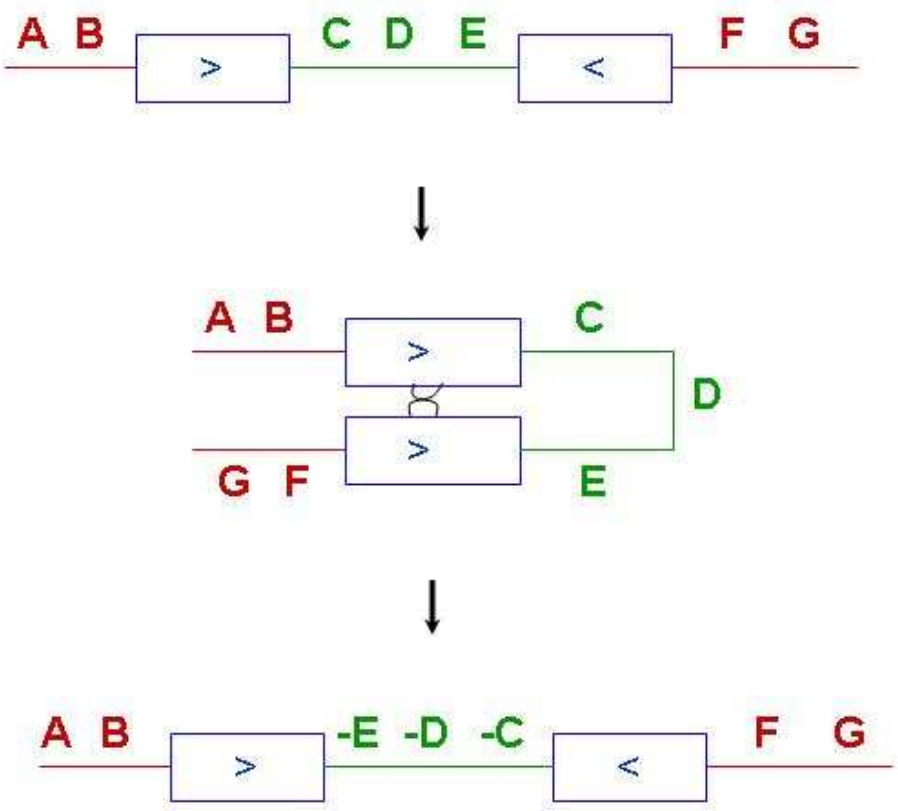


Mechanism

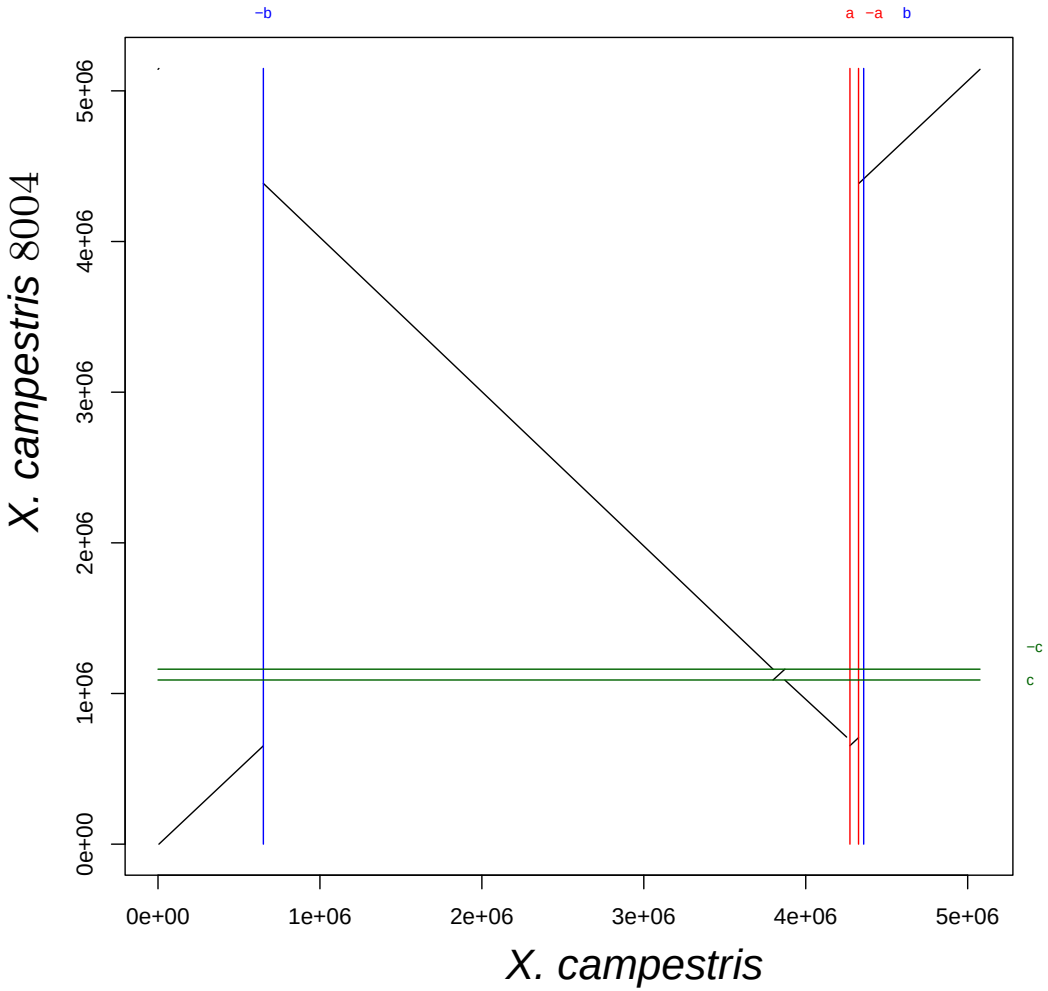
Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

Recombinations (homologous and illegitimate) are induced by repeats [Kowalczykowski et al., 1994, Smith, 1989].

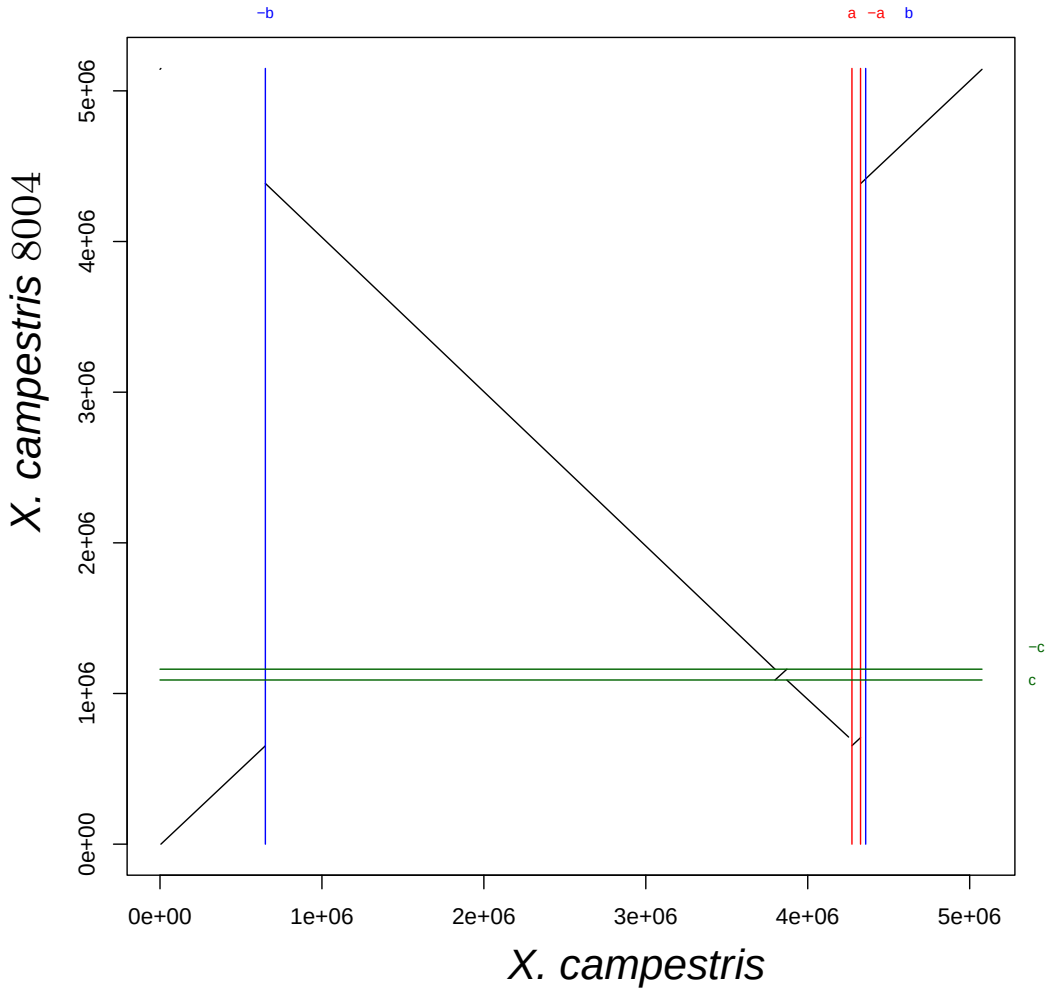


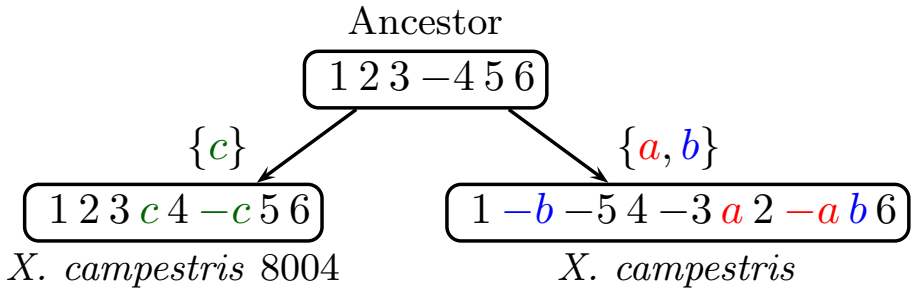
Example with repeats



MAGIC's result [Swidan et al., 2005]

Example with repeats





MAGIC's result [Swidan et al., 2005]

Properties and Assumptions

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Inversions are induced by inverted repeat pairs.

Properties and Assumptions

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Inversions are induced by inverted repeat pairs.
- Repeats correspond mostly to *selfish DNA* elements.

Properties and Assumptions

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Inversions are induced by inverted repeat pairs.
- Repeats correspond mostly to *selfish DNA* elements.
- Repeats that caused an inversion are present *only* in the affected organism.

Properties and Assumptions

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Inversions are induced by inverted repeat pairs.
- Repeats correspond mostly to *selfish DNA* elements.
- Repeats that caused an inversion are present *only* in the affected organism.
- Each repeat has a very low probability for causing an inversion that remains fixed in the population.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

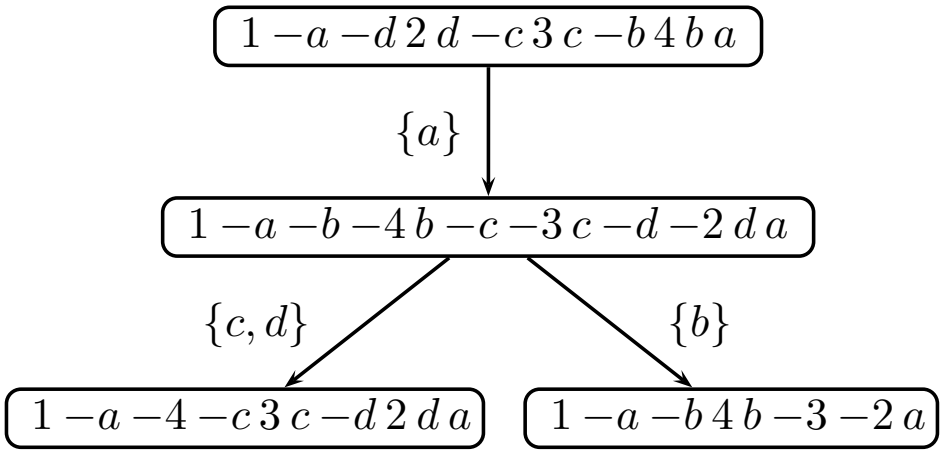
- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.

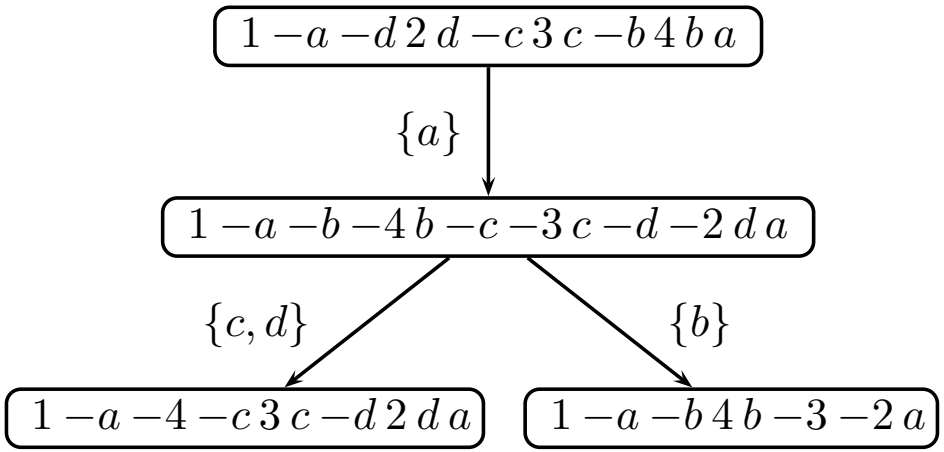


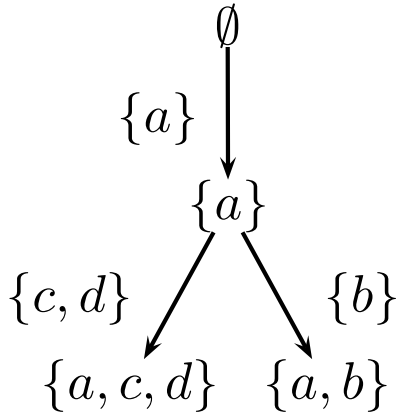
Results

Introduction
- Example
- Mechanism
- Example (cont)
- Model
- Results

Single leaf

- Single leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.
- Multiple leaf case:
  - Uniqueness proof.
  - Reconstruction in linear-time complexity.





Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Single leaf

Notation

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.

Notation

Introduction

Single leaf

❖ Notation

❖ Overview

❖ Properties

❖ Uniqueness

❖ Algorithm

❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.

Notation

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.
- *Direct repaet pair:* $-b, -b$.

Notation

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.
- *Direct repaet pair:* $-b, -b$.
- *Repeat-set:* $\mathscr{R}(s) = \{a, b\}$.

Notation

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.
- *Direct repaet pair:* $-b, -b$.
- *Repeat-set:* $\mathscr{R}(s) = \{a, b\}$.
- *Induced permutation:* $s|_N = \quad 1 \quad -4 \quad 3 \quad 2 \quad 5$.

Notation

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.
- *Direct repaet pair:* $-b, -b$.
- *Repeat-set:* $\mathscr{R}(s) = \{a, b\}$.
- *Induced permutation:* $s|_N = \quad 1 \quad -4 \quad 3 \quad 2 \quad 5$.
- *Repeat-subsequence:* $s|_R = \quad a \quad -a \quad -b \quad -b$.

Notation

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Repmaps:* $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.
- *Inverted repeat pair:* $+a, -a$.
- *Direct repaet pair:* $-b, -b$.
- *Repeat-set:* $\mathscr{R}(s) = \{a, b\}$.
- *Induced permutation:* $s|_N = \quad 1 \quad -4 \quad 3 \quad 2 \quad 5$.
- *Repeat-subsequence:* $s|_R = \quad a \quad -a \quad -b \quad -b$.
- *Breakpoint*: $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5$.

Legal reversals

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Legal reversal:*

$$s = \quad 1 \quad a \quad [-4] \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5 \ .$$

Legal reversals

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- *Legal reversal:*

$$s = \quad 1 \quad a \quad [-4] \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5 \ .$$

- *Illegal reversal:*

$$s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad [3 \quad 2] \quad -b \quad 5 \ .$$

Legal reversals

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

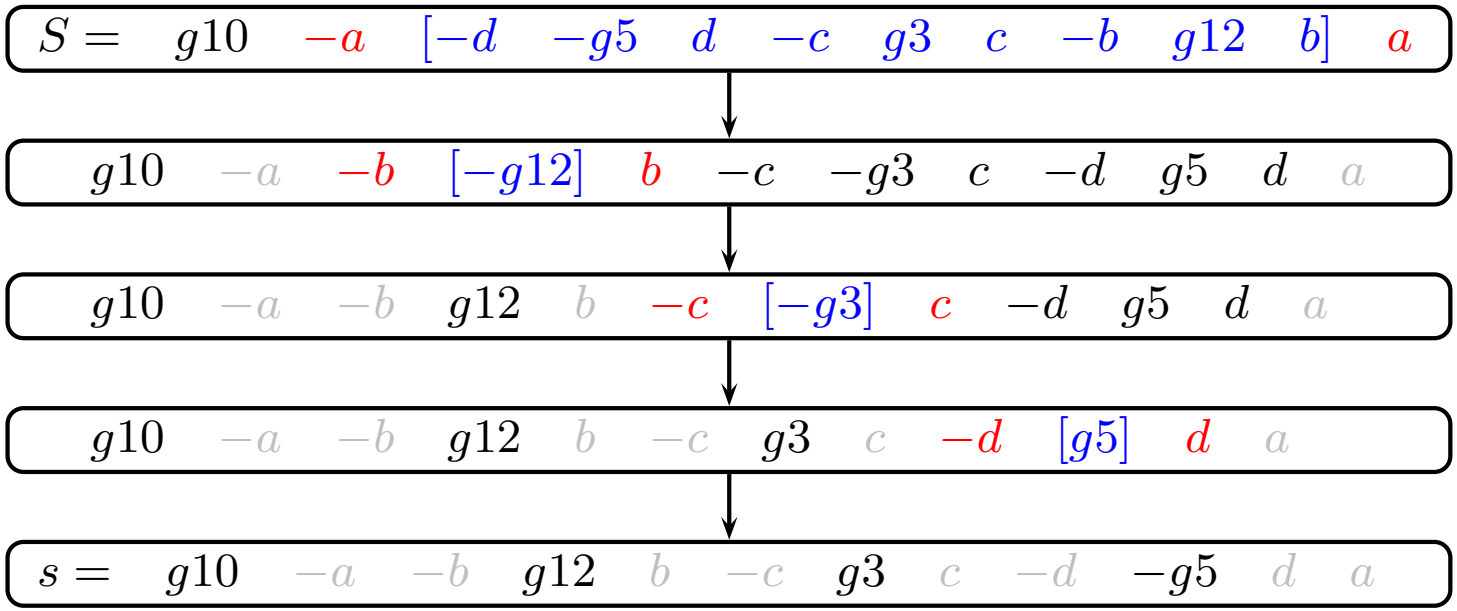
- *Legal reversal:*
  $s = \quad 1 \quad a \quad [-4] \quad -a \quad -b \quad 3 \quad 2 \quad -b \quad 5 \ .$
- *Illegal reversal:*
  $s = \quad 1 \quad a \quad -4 \quad -a \quad -b \quad [3 \quad 2] \quad -b \quad 5 \ .$
- A reversal sequence ϱ is *legal* on a repmap s if all its reversals are legal and it fulfills each repeat in $\mathscr{R}(s)$ exactly once.

Example

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

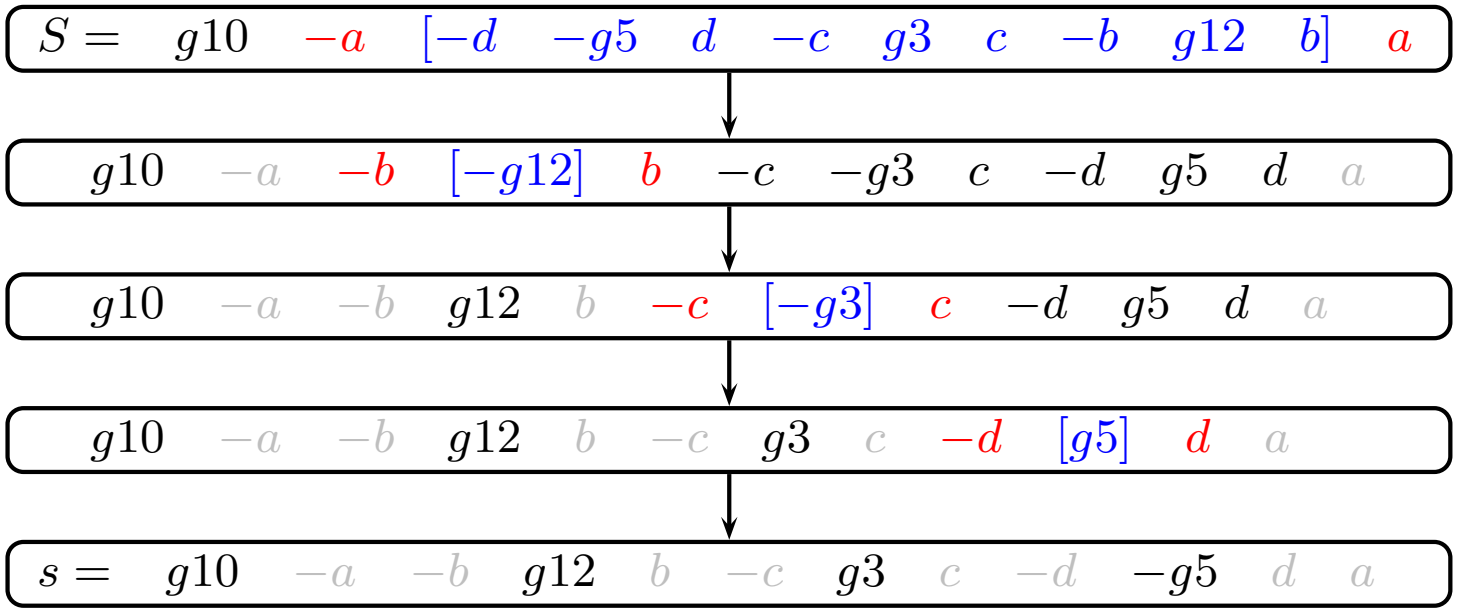


Example

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario



- Problem: Given a descendant repmap $s = S \cdot \varrho$ (S, ϱ unknown), can we reconstruct S, ϱ and are they unique?

Simplifications

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Join successive permutation elements:

$$g1 \quad g3 \quad a \quad g8 \quad b \quad -a \quad g7 \quad -g2 \quad b \quad g10$$

Simplifications

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Join successive permutation elements:

$$g1 \quad g3 \quad a \quad g8 \quad b \quad -a \quad g7 \quad -g2 \quad b \quad g10$$
$$g5 \quad a \quad g8 \quad b \quad -a \quad g6 \quad b \quad g10$$

Simplifications

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Join successive permutation elements:

  $g1 \quad g3 \quad a \quad g8 \quad b \quad -a \quad g7 \quad -g2 \quad b \quad g10$

  $g5 \quad a \quad g8 \quad b \quad -a \quad g6 \quad b \quad g10$

- What about successive repeats?

Simplifications

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Join successive permutation elements:

  $g1 \quad g3 \quad a \quad g8 \quad b \quad -a \quad g7 \quad -g2 \quad b \quad g10$

  $g5 \quad a \quad g8 \quad b \quad -a \quad g6 \quad b \quad g10$
- What about successive repeats?
- *Normalized repmaps:* contain no successive repeats

  $3 \quad a \quad 5 \quad b \quad -2 \quad -a \quad 1 \quad b \quad -4$

Proof overview

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Study the properties of normalized repmaps and legal scenarios.

Proof overview

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.

Proof overview

Introduction

Single leaf

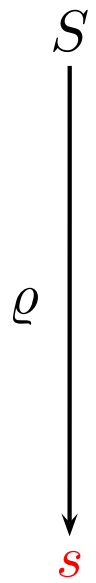
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.

Proof overview

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

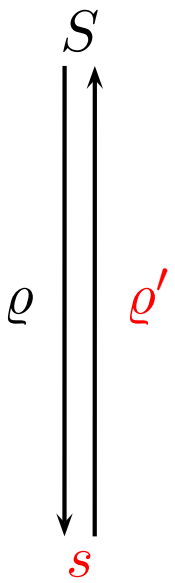
- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.



Proof overview

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

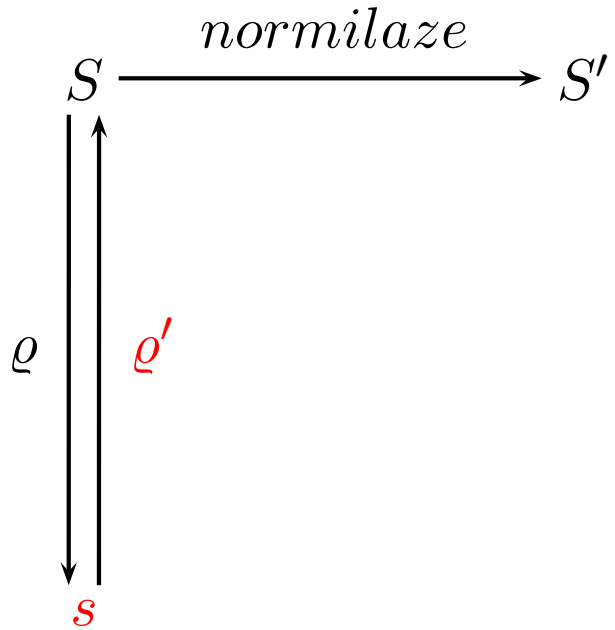
- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.



Proof overview

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

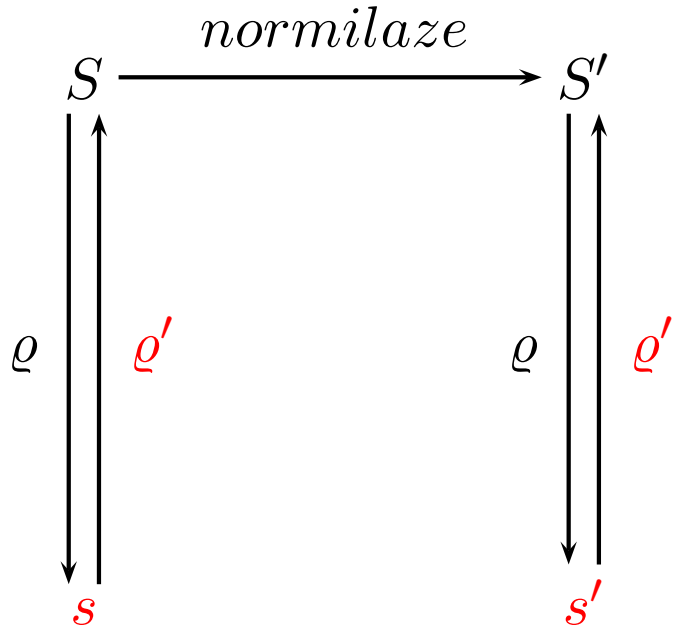
- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.



Proof overview

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

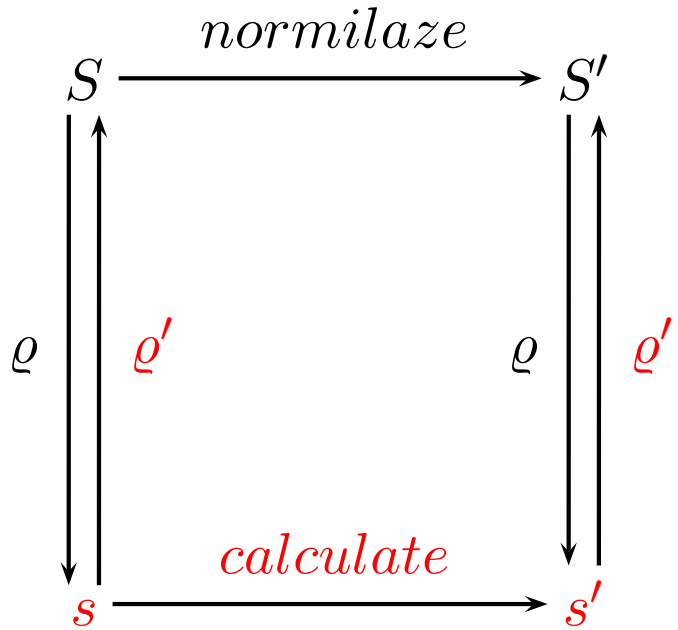
- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.



Proof overview

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Study the properties of normalized repmaps and legal scenarios.
- Prove claims for normalized repmaps.
- Transfrom general repmaps to normalized ones to finish the proof.



Properties: Surrounding contiguity

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Consider the surroundings of a repeat before and after being fulfilled.

Properties: Surrounding contiguity

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Consider the surroundings of a repeat before and after being fulfilled.
- Befor being fulfilled:

$$\cdots \quad -7 \quad b \, [\quad 8 \quad \cdots \quad 3 \quad] \, -b \quad -2 \quad \cdots$$

Properties: Surrounding contiguity

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Consider the surroundings of a repeat before and after being fulfilled.
- Befor being fulfilled:

$$\cdots \quad -7 \quad b \, [\quad 8 \quad \cdots \quad 3 \quad] \, -b \quad -2 \quad \cdots$$

- After being fulfilled:

$$\cdots \quad -7 \quad b \quad -3 \quad \cdots \quad -8 \quad -b \quad -2 \quad \cdots$$

Properties: Surrounding contiguity

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Consider the surroundings of a repeat before and after being fulfilled.
- Befor being fulfilled:

  $$\cdots \quad -7 \quad b \, [\quad 8 \quad \cdots \quad 3 \quad] \, -b \quad -2 \quad \cdots$$

- After being fulfilled:

  $$\cdots \quad -7 \quad b \quad -3 \quad \cdots \quad -8 \quad -b \quad -2 \quad \cdots$$

- The surrounding of b after being fulfilled remains fixed.

Properties: Surrounding contiguity

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Consider the surroundings of a repeat before and after being fulfilled.
- Befor being fulfilled:

  $$\cdots \quad -7 \quad b \, [\quad 8 \quad \cdots \quad 3 \quad] \, -b \quad -2 \quad \cdots$$

- After being fulfilled:

  $$\cdots \quad -7 \quad b \quad -3 \quad \cdots \quad -8 \quad -b \quad -2 \quad \cdots$$

- The surrounding of b after being fulfilled remains fixed.

Claim 1. *The surrounding of a fulfilled repeat remains fixed.*

Properties: Breakpoint elimination

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Assume $s' = S' \cdot \varrho$, where S' is sorted.

Properties: Breakpoint elimination

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

- Assume $s' = S' \cdot \varrho$, where S' is sorted.
- For example

$$s' = \quad 1 \quad -a \; [\quad -4 \quad -b \quad 3 \quad b \quad -2 \quad] \; a \quad 5$$

Properties: Breakpoint elimination

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Assume $s' = S' \cdot \varrho$, where S' is sorted.
- For example

$$s' = \quad 1 \quad -a \, [\quad -4 \quad -b \quad 3 \quad b \quad -2 \quad] \, a \quad 5$$

- Then, all legal reversals affecting s' eliminate 2 breakpoints.

Properties: Breakpoint elimination

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Assume $s' = S' \cdot \varrho$, where S' is sorted.
- For example

$$s' = \quad 1 \quad -a \, [\quad -4 \quad -b \quad 3 \quad b \quad -2 \quad] \, a \quad 5$$

- Then, all legal reversals affecting s' eliminate 2 breakpoints.

Lemma 2. *A legal reversal affecting s' eliminates two breakpoints.*

Properties: Breakpoint elimination

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Assume $s' = S' \cdot \varrho$, where S' is sorted.
- For example
  $$s' = \quad 1 \quad -a\,[\quad -4 \quad -b \quad 3 \quad b \quad -2 \quad]\,a \quad 5$$
- Then, all legal reversals affecting s' eliminate 2 breakpoints.

Lemma 2. *A legal reversal affecting s' eliminates two breakpoints.*

By induction, we get the following:

Corollary 3. *All sequential legal reversals affecting s' eliminate two breakpoints.*

Uniqueness for normalized repmaps

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

Lemma 4. *Assuming $s' = S' \cdot \varrho$, all legal scenarios affecting s' result in S'.*

Uniqueness for normalized repmaps

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

Lemma 4. *Assuming $s' = S' \cdot \varrho$, all legal scenarios affecting s' result in S'.*

Corollary 5. *A legal scenario is an SBR scenario.*

Uniqueness

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Theorem 6 (Ancestor Uniqueness)**.** *Assuming* $s = S \cdot \varrho$*, where* S *is an arbitrary repmap, all legal scenarios affecting* s *result in* S*.*

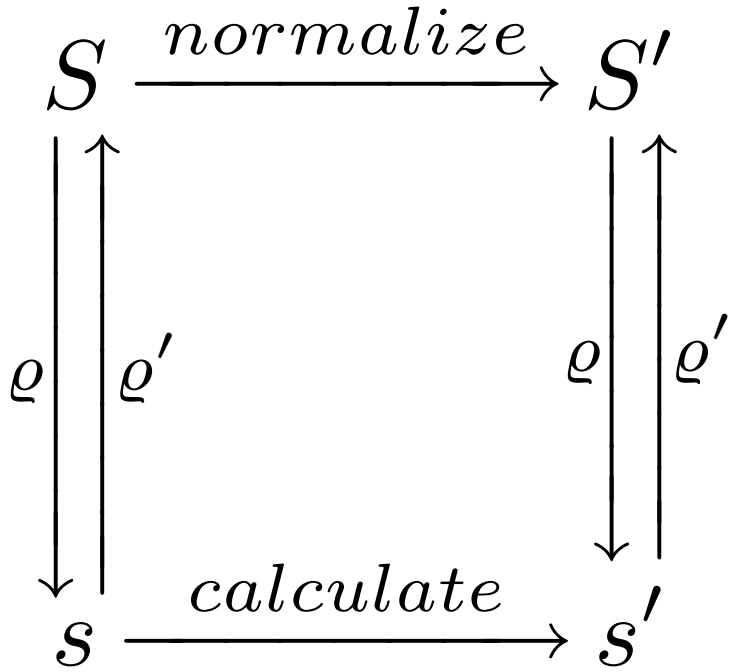
Uniqueness

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Theorem 6 (Ancestor Uniqueness)**.** *Assuming $s = S \cdot \varrho$, where S is an arbitrary repmap, all legal scenarios affecting s result in S.*



Calculating the ancestor

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.
   - This can be done, *e.g.*, by solving the *overlap graph* defined over the repeat pair arcs [Kaplan, Shamir, and Tarjan, 1997].

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.
   - This can be done, *e.g.*, by solving the *overlap graph* defined over the repeat pair arcs [Kaplan, Shamir, and Tarjan, 1997].
   - Best known performance has quadratic time-complexity.

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.
   - This can be done, *e.g.*, by solving the *overlap graph* defined over the repeat pair arcs [Kaplan, Shamir, and Tarjan, 1997].
   - Best known performance has quadratic time-complexity.
2. Reconstruct the ancestor directly, without calculating a legal scenario.

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.
   - This can be done, *e.g.*, by solving the *overlap graph* defined over the repeat pair arcs [Kaplan, Shamir, and Tarjan, 1997].
   - Best known performance has quadratic time-complexity.
2. Reconstruct the ancestor directly, without calculating a legal scenario.
   - Use the locality of the repeats' surroundings?

Calculating the ancestor

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

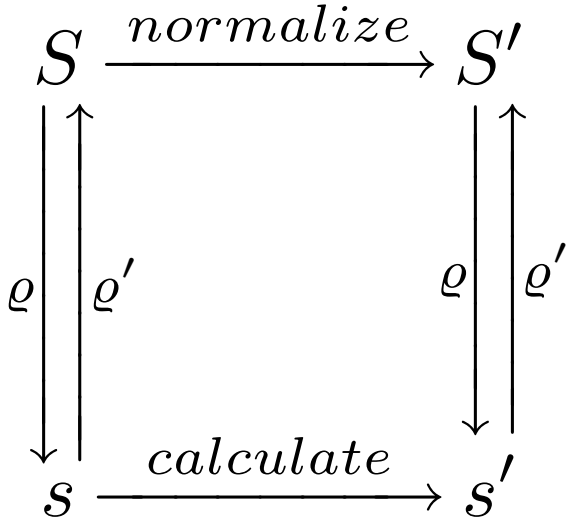
Reconstructing the ancestor can be achieved using different strategies.

1. Reconstruct a legal scenario and apply it to s to reconstruct the ancestor.
   - This can be done, *e.g.*, by solving the *overlap graph* defined over the repeat pair arcs [Kaplan, Shamir, and Tarjan, 1997].
   - Best known performance has quadratic time-complexity.
2. Reconstruct the ancestor directly, without calculating a legal scenario.
   - Use the locality of the repeats' surroundings?
   - Results in a linear-time algorithm.

Strategy

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

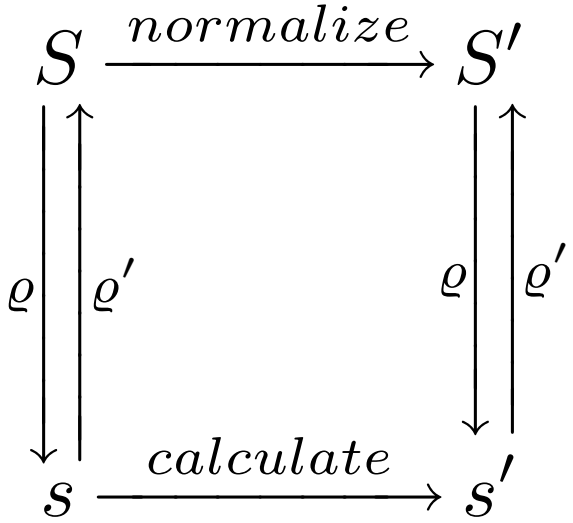
- Given that $s = S \cdot \varrho$, for an arbitrary repmap S, we want to compute $s' = S' \cdot \varrho$, where S' is the sorted normalization of S.



Strategy

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

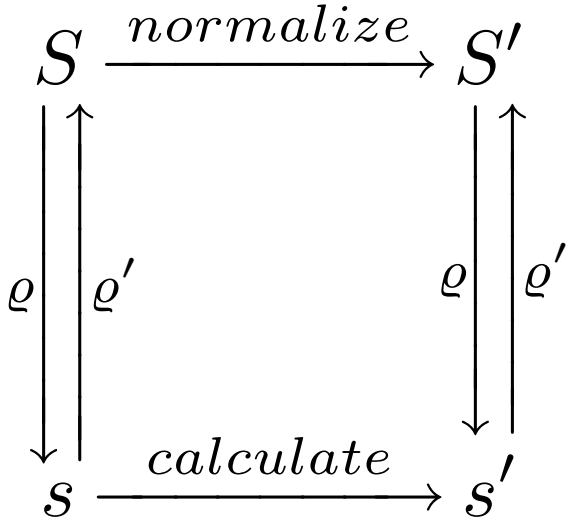
- Given that $s = S \cdot \varrho$, for an arbitrary repmap S, we want to compute $s' = S' \cdot \varrho$, where S' is the sorted normalization of S.
- Then, given s', we know that its ancestor S' is sorted.



Strategy

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

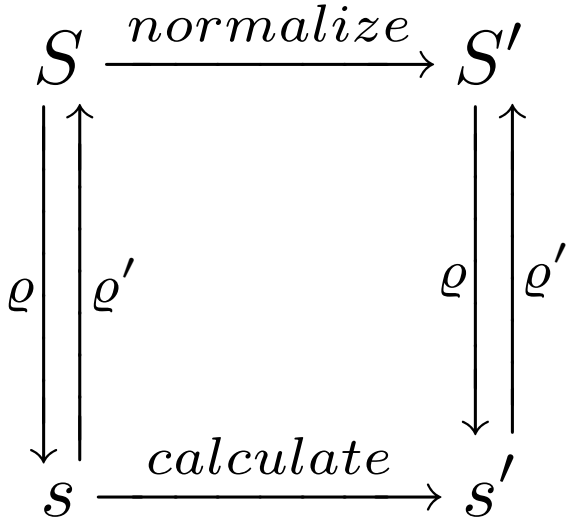
- Given that $s = S \cdot \varrho$, for an arbitrary repmap S, we want to compute $s' = S' \cdot \varrho$, where S' is the sorted normalization of S.
- Then, given s', we know that its ancestor S' is sorted.
- By using the renaming from s to s', we can rename S' to S in linear-time.



Strategy

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

- Given that $s = S \cdot \varrho$, for an arbitrary repmap S, we want to compute $s' = S' \cdot \varrho$, where S' is the sorted normalization of S.
- Then, given s', we know that its ancestor S' is sorted.
- By using the renaming from s to s', we can rename S' to S in linear-time.
- How to compute s' from s?



Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

$$s' = 1 \ -a \ ? \ -b \ ? \ b \ ? \ -c \ ? \ c \ ? \ -d \ ? \ d \ ? \ a \ 9$$

Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

$$s' = 1 \quad -a \quad ? \quad -b \quad ? \quad b \quad ? \quad -c \quad ? \quad c \quad ? \quad -d \quad ? \quad d \quad ? \quad a \quad 9$$

Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

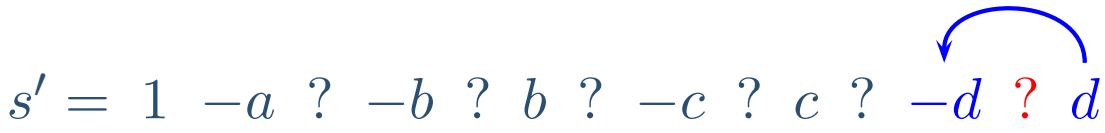
$$s' = 1 \quad -a \quad ? \quad -b \quad ? \quad b \quad ? \quad -c \quad ? \quad c \quad ? \quad -d \quad ? \quad d \quad -2 \quad a \quad 9$$

Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

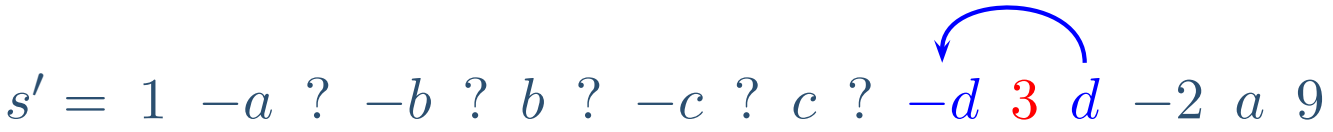


Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

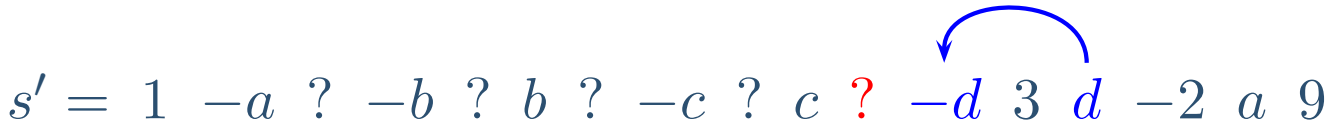


Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

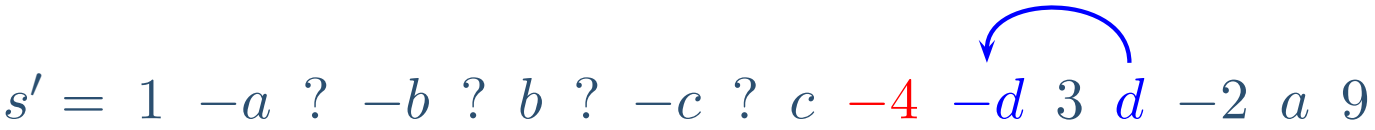


Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:



Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

$$s' = \; 1 \; -a \; -8 \; -b \; 7 \; b \; -6 \; -c \; 5 \; c \; -4 \; -d \; 3 \; d \; -2 \; a \; 9$$

Normalizing s

Example:

$$s = \quad g10 \quad -a \quad -b \quad g12 \quad b \quad -c \quad g3 \quad c \quad -d \quad -g5 \quad d \quad a$$

Normalization:

$$s' = 1 \ -a \ -8 \ -b \ 7 \ b \ -6 \ -c \ 5 \ c \ -4 \ -d \ 3 \ d \ -2 \ a \ 9$$

Lemma 7. *The normalized repmap s' can be computed iteratively in linear-time from the repmap s.*

Reconstruction time-complexity

Introduction

Single leaf

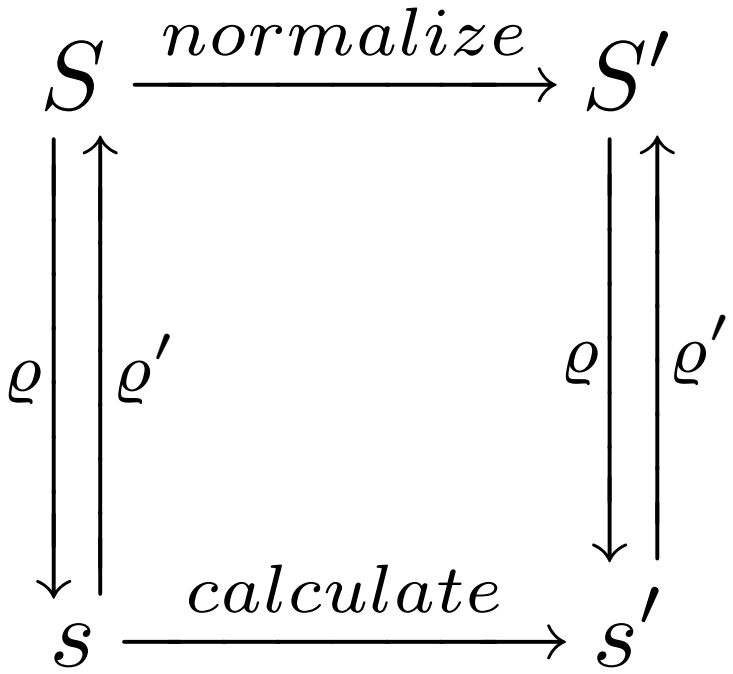
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Theorem 8 (Time-complexity)**.** *The ancestral repmap S can be reconstructed from s in linear-time.*

Reconstruction time-complexity

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Theorem 8 (Time-complexity)**.** *The ancestral repmap S can be reconstructed from s in linear-time.*



The scenario

Introduction
Single leaf
❖ Notation
❖ Overview
❖ Properties
❖ Uniqueness
❖ Algorithm
❖ Scenario

Theorem 9 (Scenario non-uniqueness)**.** *The scenarios are not uniquely determined by the repmap* s*.*

The scenario

Introduction
Single leaf
- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

Theorem 9 (Scenario non-uniqueness)**.** *The scenarios are not uniquely determined by the repmap* s*.*

Theorem 10 (Time-complexity)**.** *A legal scenario of a repmap containing* m *repeats can be calculated in* $O(m\sqrt{m \log m})$ *using [Tannier and Sagot, 2004].*

Introduction

Single leaf

- Notation
- Overview
- Properties
- Uniqueness
- Algorithm
- Scenario

End

Introduction

Single leaf

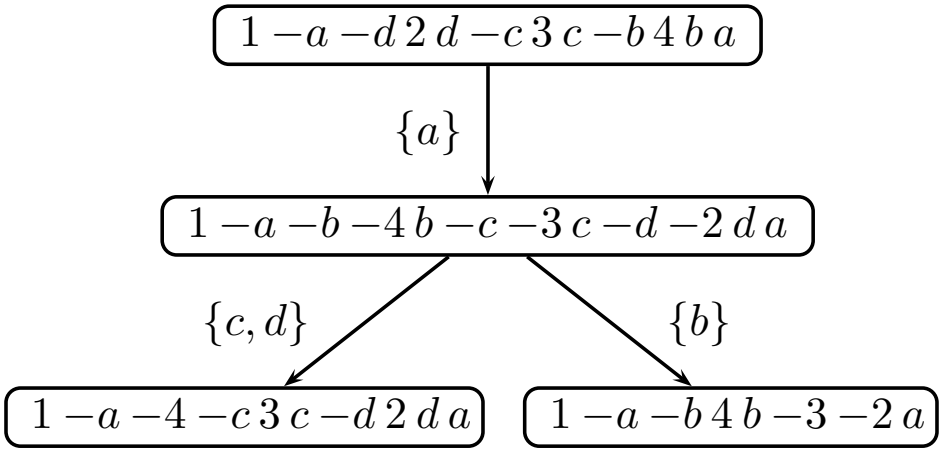
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Multiple leafs

Notation

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Repeat-annotated phylogenetic tree (RAPT):



Notation

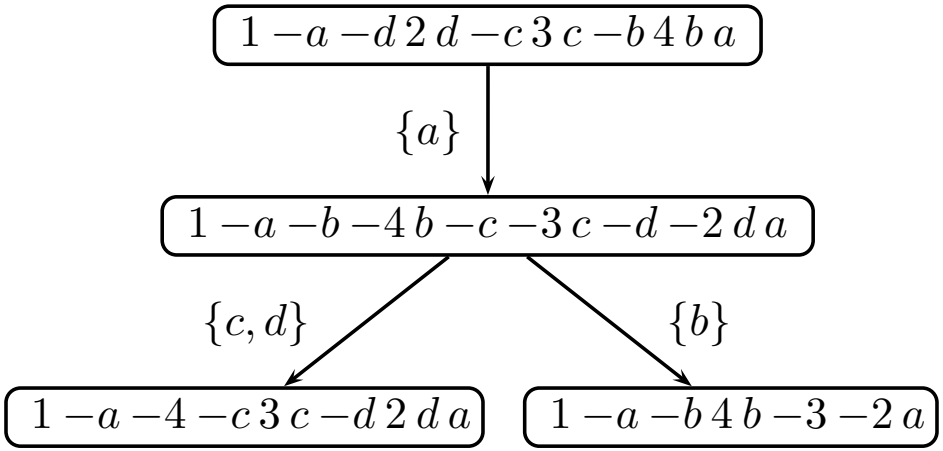
Introduction

Single leaf

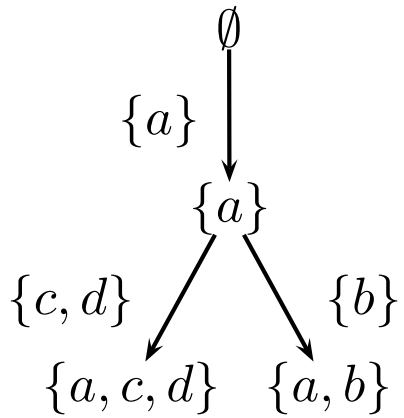
Multiple leafs

- Notation
- properties
- Uniqueness
- Algorithm

Repeat-annotated phylogenetic tree (RAPT):



Set-trie:



Monotonic collections

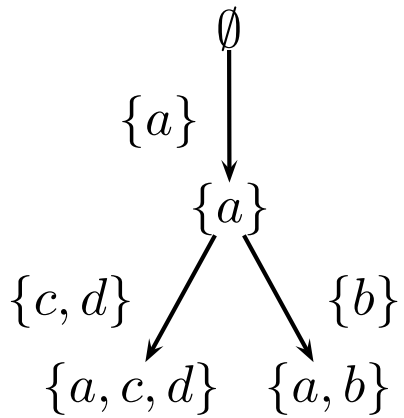
Introduction

Single leaf

Multiple leafs

- Notation
- properties
- Uniqueness
- Algorithm

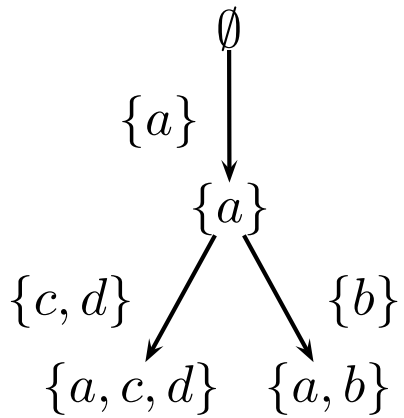
Set-trie:



Monotonic collections

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Set-trie:



Definition 11 (Monotonic Set Collection)**.** A set collection $\mathscr{A}$ is *monotonic* if $\forall A, B, C \in \mathscr{A}$, either $A \cap B \subseteq A \cap C$ or $A \cap C \subseteq A \cap B$.

Set-tries and monotonic collections

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Lemma 12. *Let $\mathscr{A}$ be a finite collection of finite subsets. There exists a set-trie over $\mathscr{A}$ iff $\mathscr{A}$ is monotonic.*

Set-tries and monotonic collections

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Lemma 12. *Let $\mathscr{A}$ be a finite collection of finite subsets. There exists a set-trie over $\mathscr{A}$ iff $\mathscr{A}$ is monotonic.*

Proof. The direction $\Rightarrow$ is trivial. The other direction ($\Leftarrow$) results from the reconstruction algorithm. □

Set-tries and monotonic collections

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Lemma 12. *Let $\mathscr{A}$ be a finite collection of finite subsets. There exists a set-trie over $\mathscr{A}$ iff $\mathscr{A}$ is monotonic.*

Proof. The direction $\Rightarrow$ is trivial. The other direction ($\Leftarrow$) results from the reconstruction algorithm. □

Lemma 13 (Set-trie uniqueness)**.** *Let $\mathscr{A}$ be a monotonic collection. There exists a unique set-trie over $\mathscr{A}$.*

Uniqueness

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 14 (RAPT uniqness)**.** *The leaf repmaps L uniquely determine the underlying RAPT up to (and not including) repeats in the inner nodes.*

Uniqueness

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 14 (RAPT uniqness)**.** *The leaf repmaps L uniquely determine the underlying RAPT up to (and not including) repeats in the inner nodes.*

Proof.

- The repeat-sets of the leaf repmaps are a monotonic collection.

Uniqueness

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 14 (RAPT uniqness)**.** *The leaf repmaps L uniquely determine the underlying RAPT up to (and not including) repeats in the inner nodes.*

Proof.

- The repeat-sets of the leaf repmaps are a monotonic collection.
- Hence, they uniquely determine the underlying RAPT topology and edge labels.

Uniqueness

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 14 (RAPT uniqness)**.** *The leaf repmaps L uniquely determine the underlying RAPT up to (and not including) repeats in the inner nodes.*

Proof.

- The repeat-sets of the leaf repmaps are a monotonic collection.
- Hence, they uniquely determine the underlying RAPT topology and edge labels.
- The edge labels, in turn, uniquely determine the inner node repmaps up to their repeats.

□

Reconstruction algorithm

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Reconstructing a set-trie from a monotonic collection:

- Based on incremental set insertions.

Reconstruction algorithm

Introduction
Single leaf
Multiple leafs
❖ Notation
❖ properties
❖ Uniqueness
❖ Algorithm

Reconstructing a set-trie from a monotonic collection:

- Based on incremental set insertions.
- Inserting a set involves two operations:

Reconstruction algorithm

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Reconstructing a set-trie from a monotonic collection:

- Based on incremental set insertions.
- Inserting a set involves two operations:
  - Finding the insertion edge.

Reconstruction algorithm

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

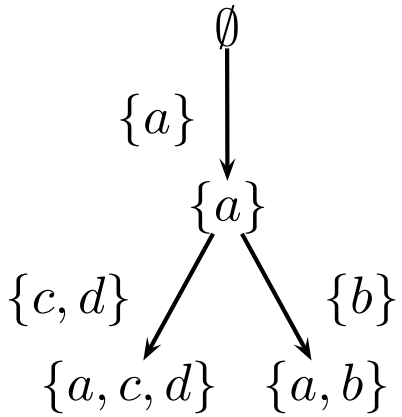
Reconstructing a set-trie from a monotonic collection:

- Based on incremental set insertions.
- Inserting a set involves two operations:
  - Finding the insertion edge.
  - Spliting the edge and modifying the tree to insert the set.

Example

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

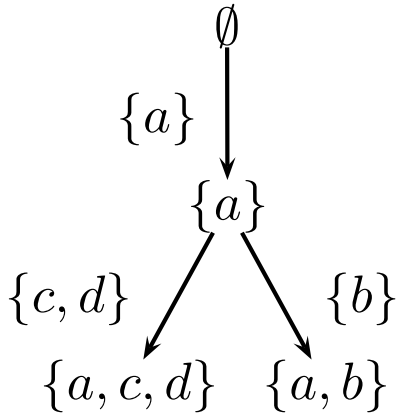
- Assume we want to insert $A = \{a, c, h\}$ in



Example

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Assume we want to insert $A = \{a, c, h\}$ in

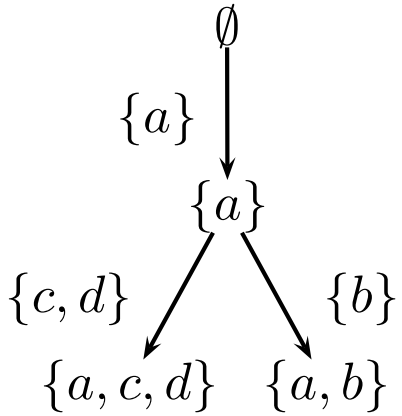


- Which edge should we cut?

Example

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Assume we want to insert $A = \{a, c, h\}$ in

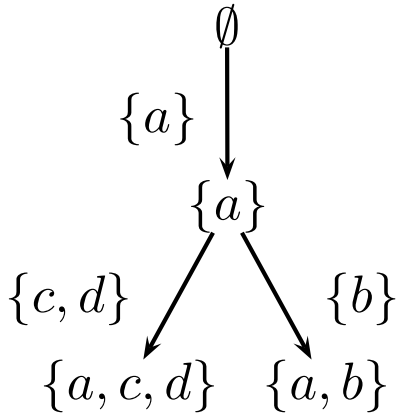


- Which edge should we cut? The last edge (e_{tail}) whose label intersects with A.

Example

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Assume we want to insert $A = \{a, c, h\}$ in

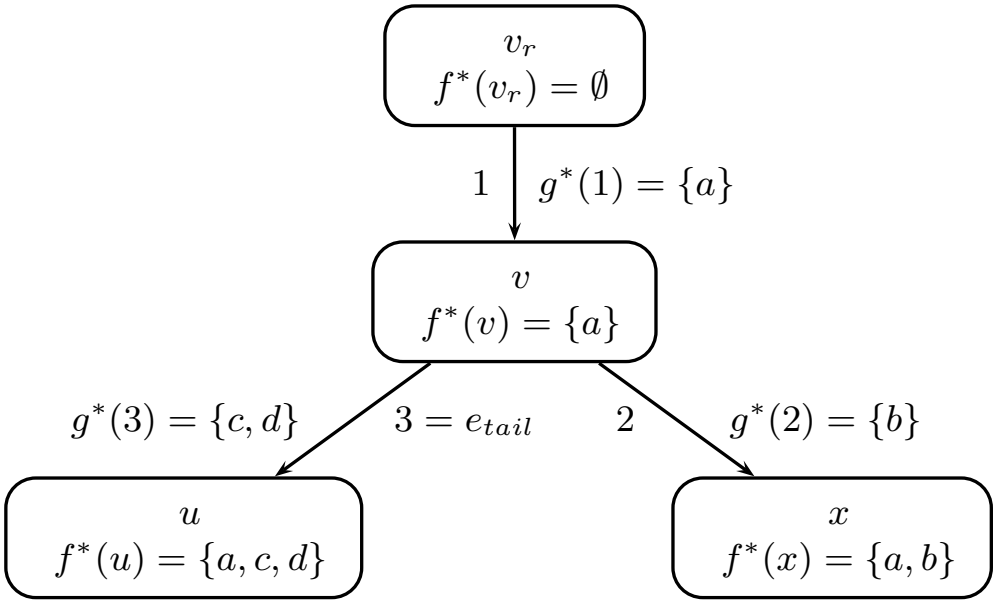


- Which edge should we cut? The last edge (e_{tail}) whose label intersects with A.
- How to efficiently find this edge?

Efficient find

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Where to insert $\{a, c, h\}$?



Efficient find

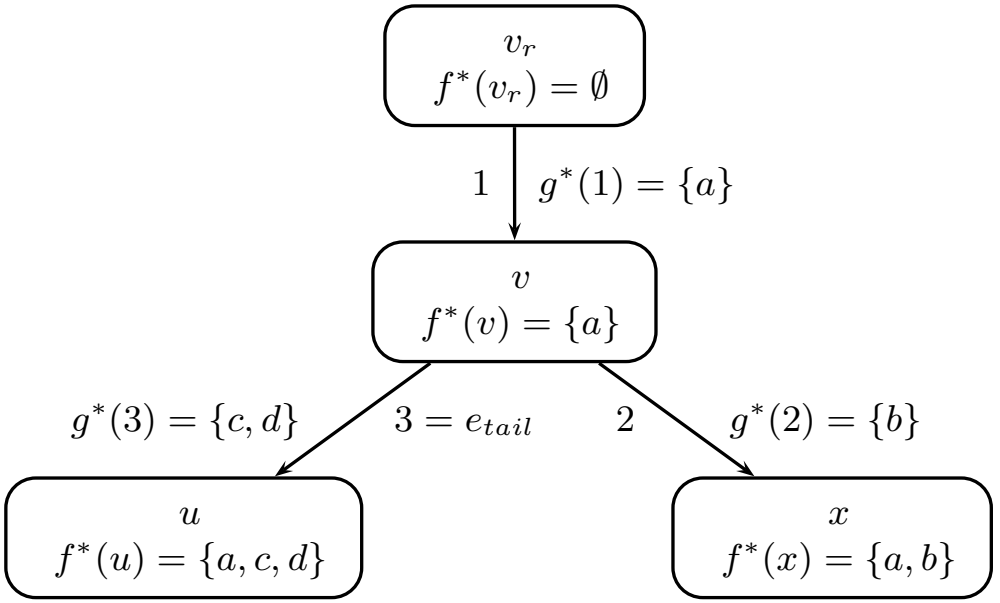
Introduction

Single leaf

Multiple leafs

- Notation
- properties
- Uniqueness
- Algorithm

- Where to insert $\{a, c, h\}$?



- For each element keep track of its location in the trie:

	a	b	c	d	e	h
e_a	**1**	2	**3**	3	null	null
e'_a	**null**	1	**1**	1	null	null

Efficient find (cont)

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Scan the elements of A and mark e'_a for each $a \in A$.

Efficient find (cont)

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Scan the elements of A and mark e'_a for each $a \in A$.
- Scane A again, and choose the edge that was not marked.

Efficient find (cont)

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Scan the elements of A and mark e'_a for each $a \in A$.
- Scane A again, and choose the edge that was not marked.
- This edge is e_{tail}.

Efficient find (cont)

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Scan the elements of A and mark e'_a for each $a \in A$.
- Scane A again, and choose the edge that was not marked.
- This edge is e_{tail}.
- Hence, the time-complexity of find is linear $O(A)$.

Efficient find (cont)

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

- Scan the elements of A and mark e'_a for each $a \in A$.
- Scane A again, and choose the edge that was not marked.
- This edge is e_{tail}.
- Hence, the time-complexity of find is linear $O(A)$.
- Split can be implemented in linear-time using similar ideas.

Time-complexity

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 15 (Set-trie time-complexity)**.** *Given a monotonic collection $\mathscr{A}$, a set-trie over $\mathscr{A}$ can be constructed in linear-time ($\Theta(|\mathscr{A}|)$).*

Time-complexity

Introduction
Single leaf
Multiple leafs
- Notation
- properties
- Uniqueness
- Algorithm

Theorem 15 (Set-trie time-complexity)**.** *Given a monotonic collection $\mathscr{A}$, a set-trie over $\mathscr{A}$ can be constructed in linear-time ($\Theta(|\mathscr{A}|)$).*

Theorem 16 (RAPT time-complexity)**.** *Given the leaf repmaps L of an unknown RAPT, reconstructing the RAPT (up to repeats in the inner nodes) can be done in linear-time ($\Theta(|L|)$).*