Parameterized Algorithms

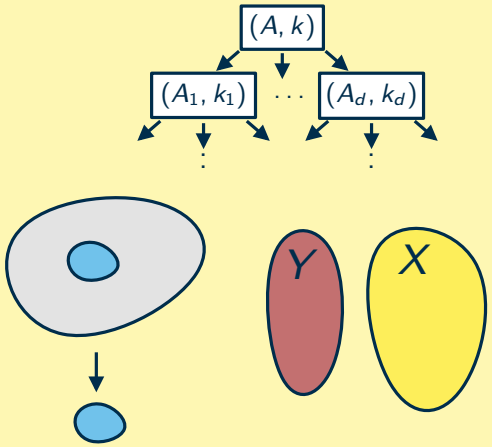
Kernelization: Similar Trees

Thomas Bläsius

Content

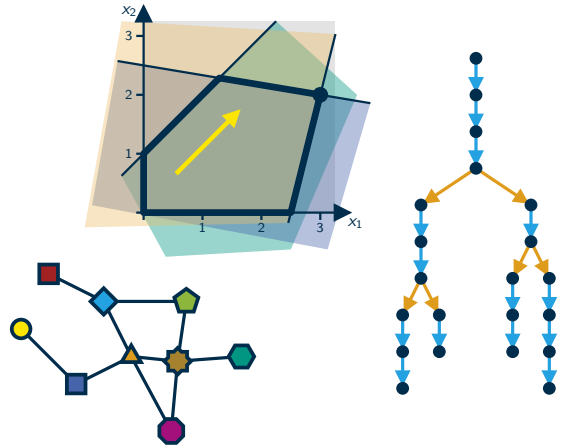
Basic toolbox

- bounded search trees
- kernelization
- iterative compression



Extended toolbox

- linear programs
- branch-and-reduce
- color coding

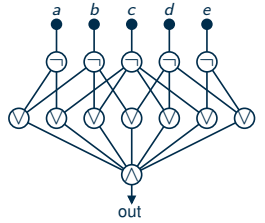


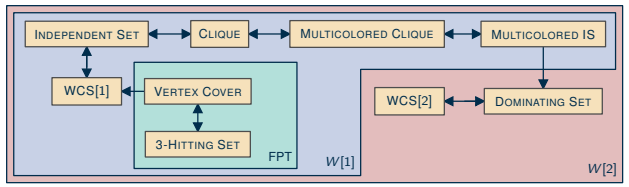
Tree width

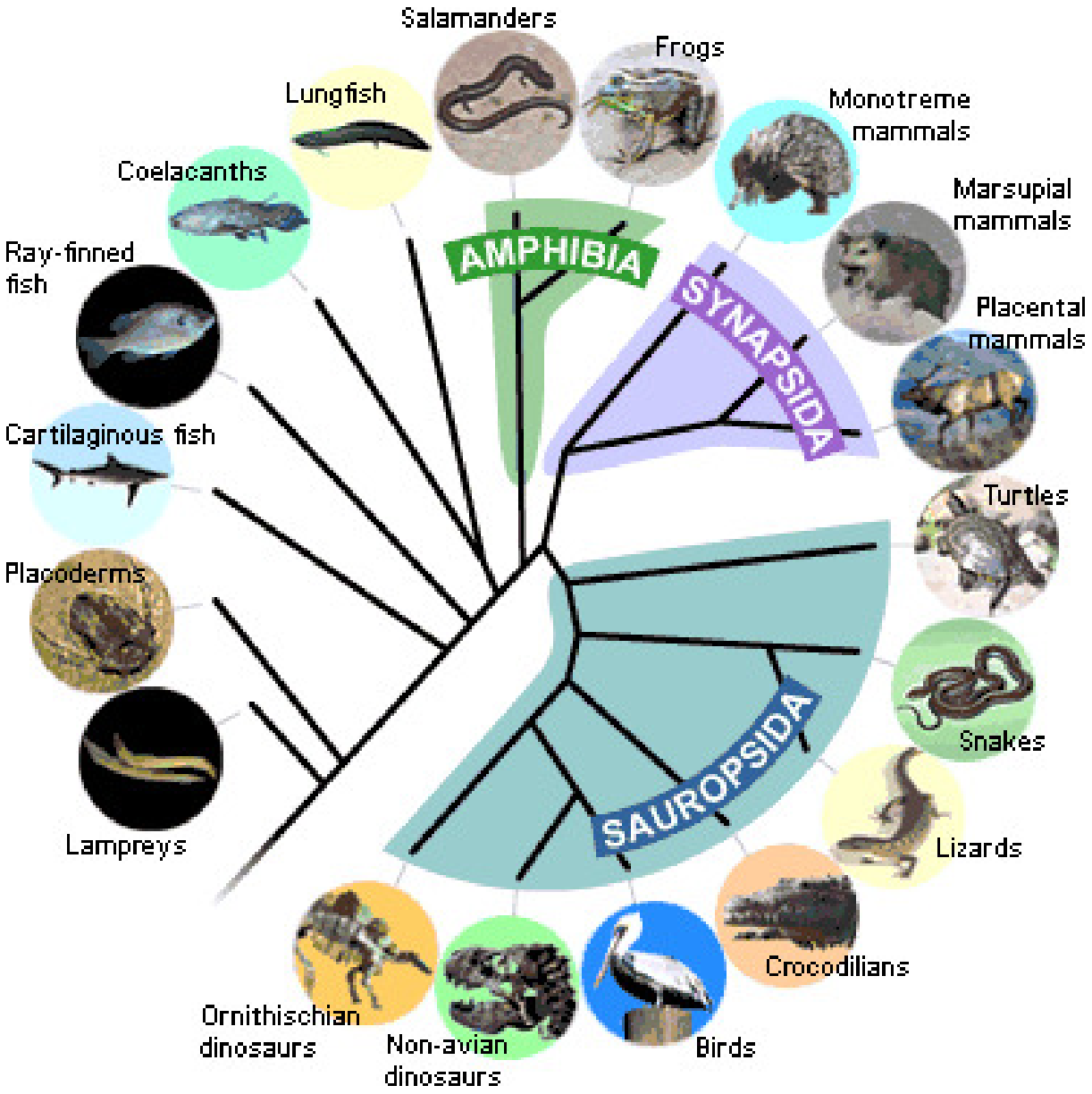
- dynamic programming
- chordal and planar graphs
- Courcelle's theorem

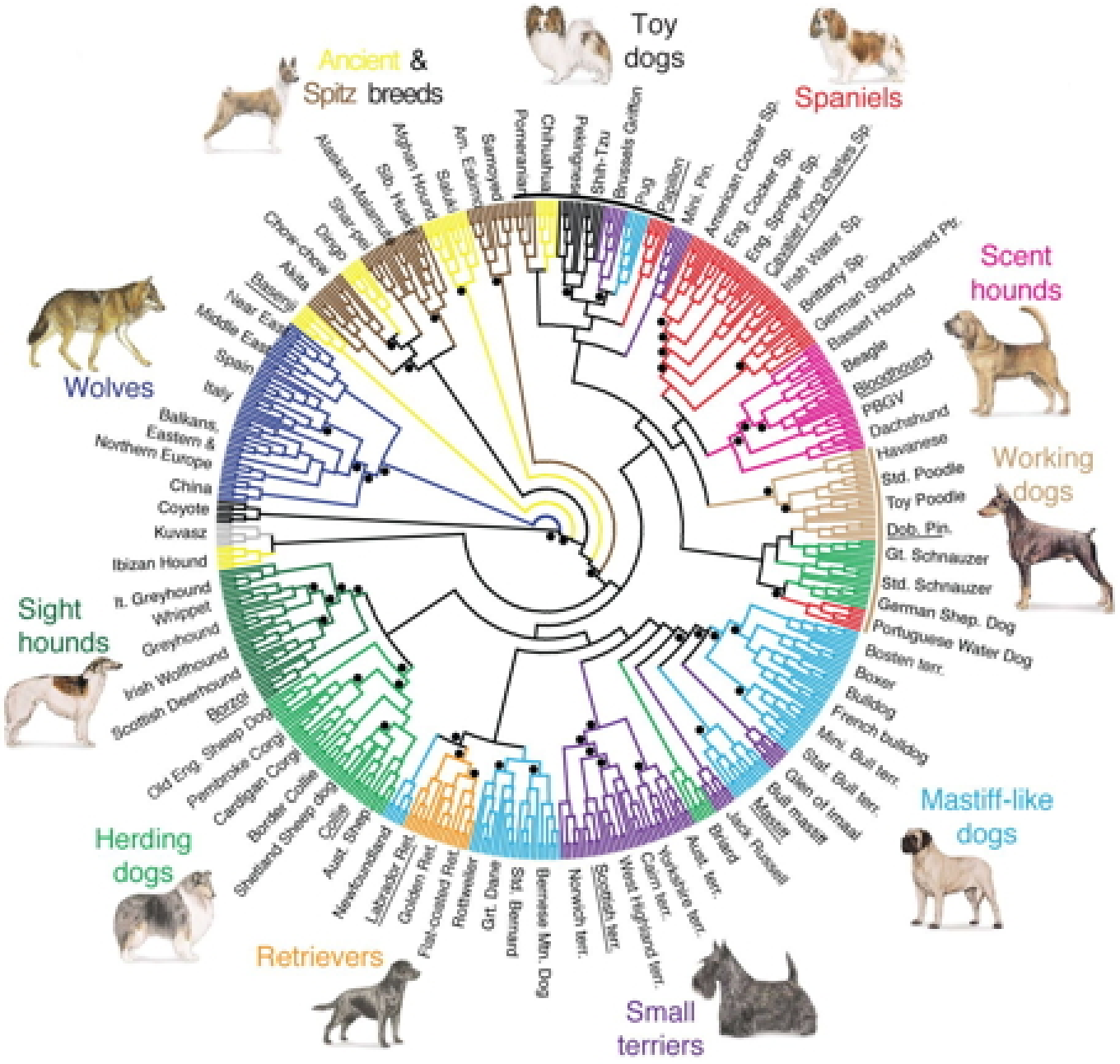
Lower bonds

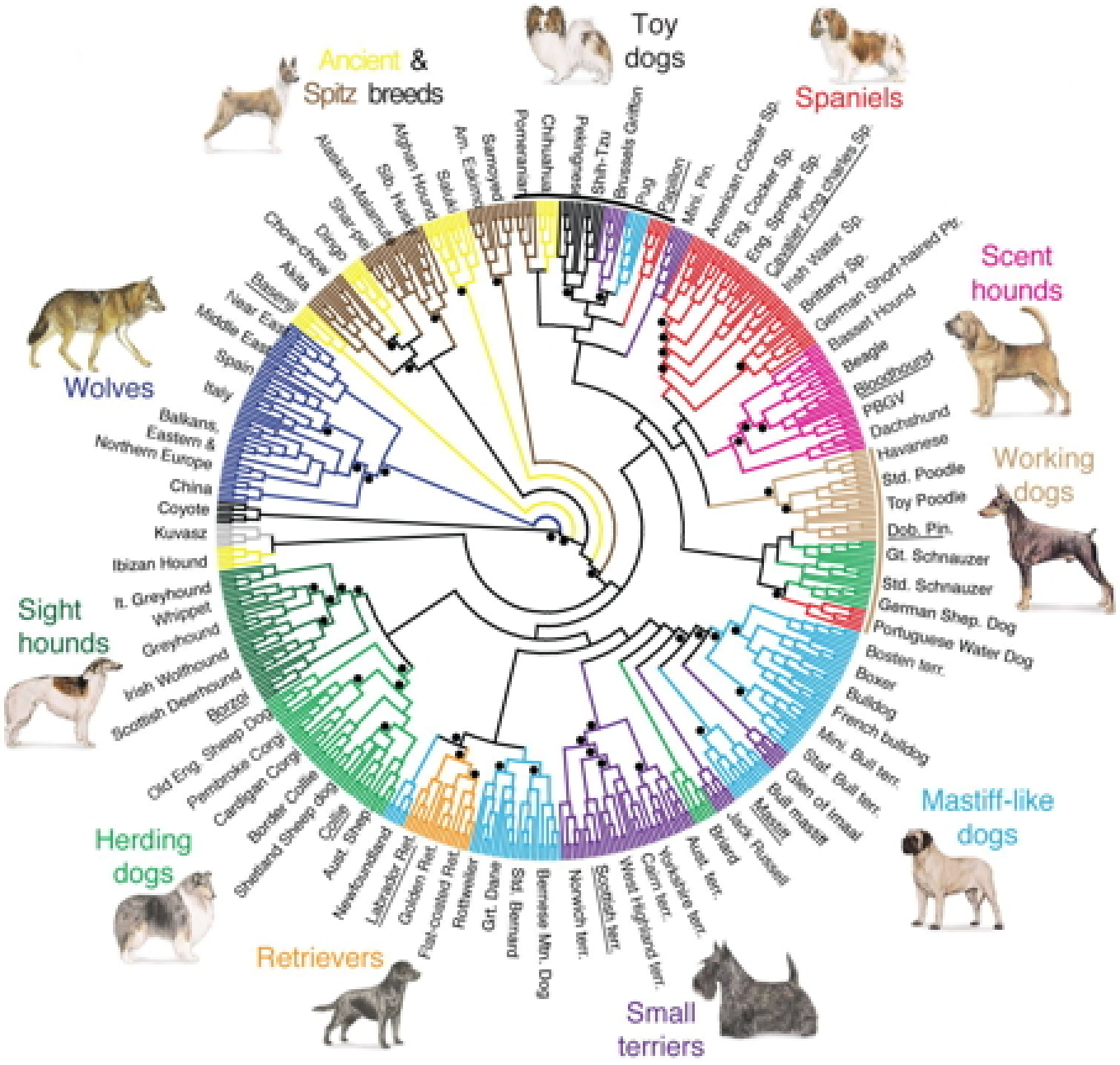
- kernel lower bounds
- parameterized reductions
- circuits and the W-hierarchy
- ETH and SETH

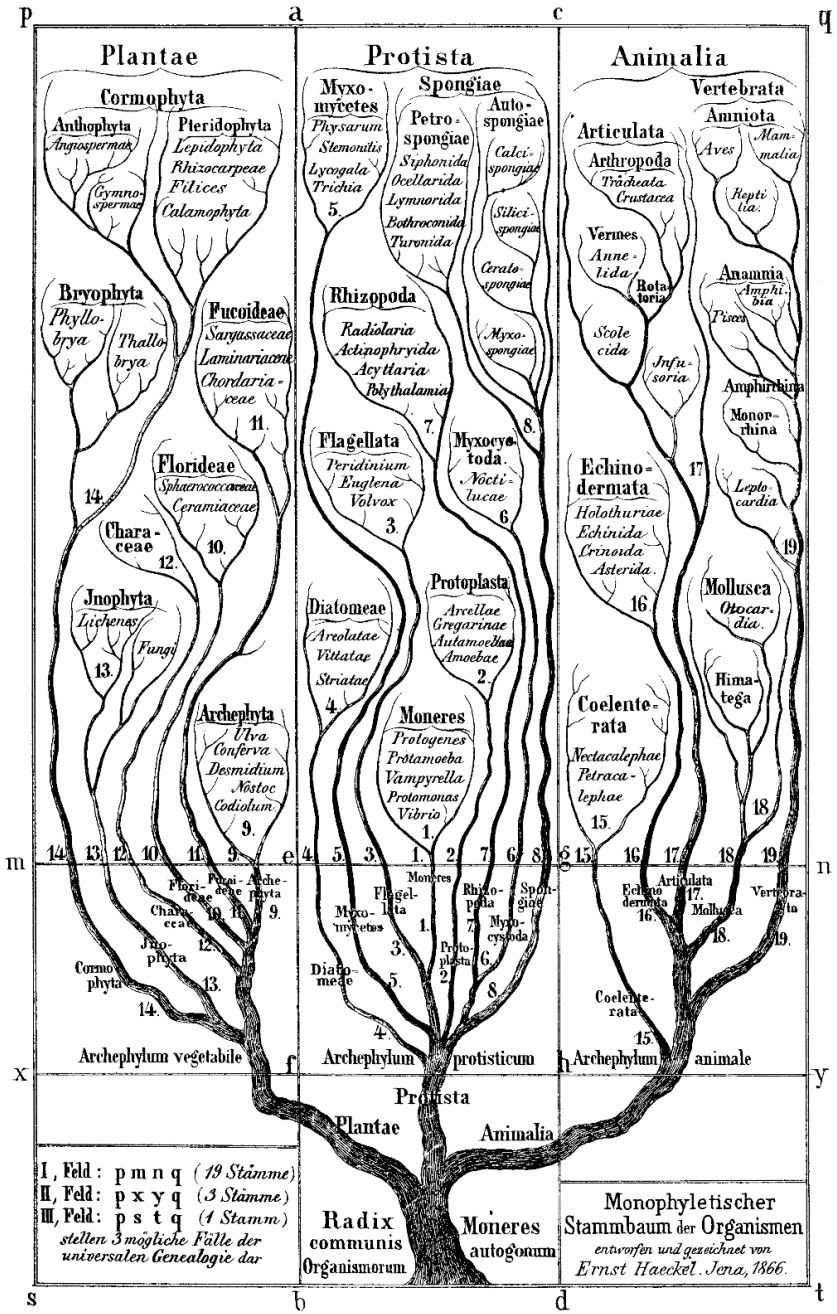


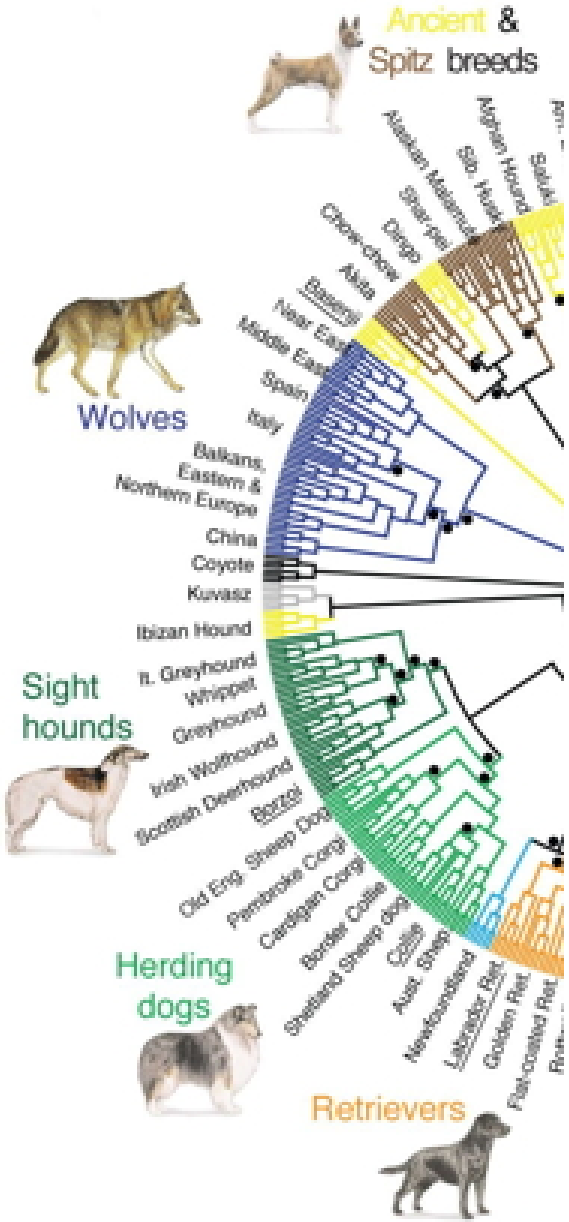


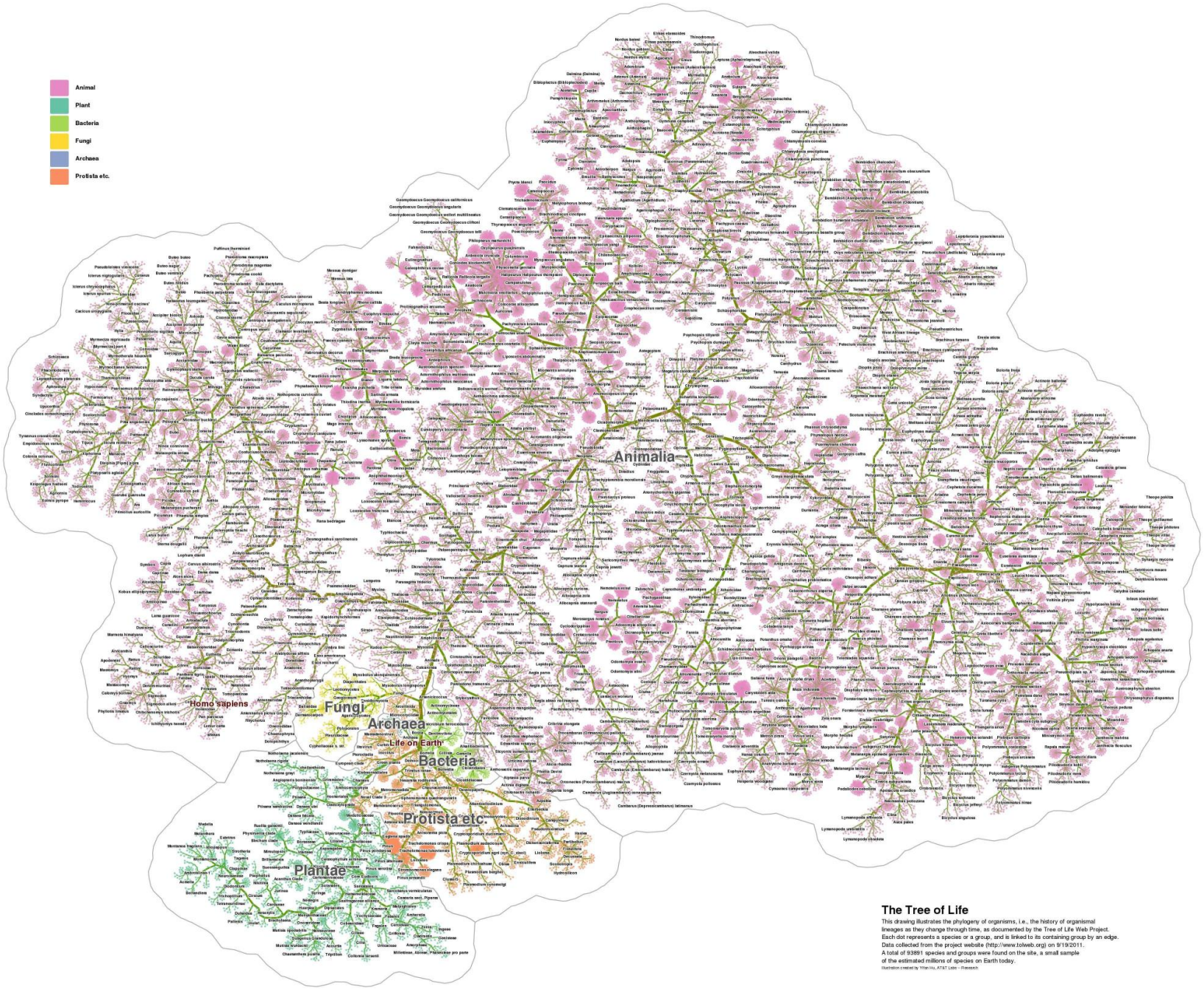

Salamanders
Frogs
Lungfish
Monotreme mammals
Coelacanths
Marsupial mammals
AMPHIBIA
SYNAPSIDA
Ray-finned fish
Placental mammals
Cartilaginous fish
Turtles
Placoderms
SAUROPSIDA
Snakes
Lizards
Lampreys
Crocodilians
Ornithischian dinosaurs
Non-avian dinosaurs
Birds


Toy dogs
Ancient & Spitz breeds
Spaniels
Scent hounds
Working dogs
Mastiff-like dogs
Small terriers
Retrievers
Herding dogs
Sight hounds
Wolves
Pomeranian
Chihuahua
Pekingese
Shih-Tzu
Brussels Griffon
Pug
Papillon
Min. Pin.
American Cocker Sp.
Eng. Cocker Sp.
Eng. Springer Sp.
Cavalier King Charles Sp.
Irish Water Sp.
Brittany Sp.
German Short-haired Pg.
Basset Hound
Beagle
Bloodhound
PBGV
Dachshund
Havanese
Std. Poodle
Toy Poodle
Dob. Pin.
Gt. Schnauzer
Std. Schnauzer
German Shep. Dog
Portuguese Water Dog
Boston terr.
Boxer
Bulldog
French bulldog
Mins. Bull terr.
Stds. Bull terr.
Glen of Imaal
Bull mastiff
Mastiff
Jack Russell
Brard
Aust. terr.
Yorkshire terr.
Cairn terr.
West Highland terr.
Scottish terr.
Norwich terr.
Bernese Mtn. Dog
Std. Bernard
Grt. Dane
Rottweiler
Flat-coated Ret.
Golden Ret.
Labrador Ret.
Newfoundland
Aust. Shep.
Collie
Shetland Sheep dog
Border Collie
Cardigan Corgi
Pembroke Corgi
Old Eng. Sheep Dog
Bergi
Scottish Deerhound
Irish Wolfhound
Greyhound
Whippet
It. Greyhound
Ibizan Hound
Kuvasz
Coyote
China
Balkans, Eastern & Northern Europe
Italy
Spain
Middle East
Near East
Basenji
Akita
Chow-chow
Dingo
Shar-pei
Alaskan Malamute
Sib. Husky
Afghan Hound
Saluki
Am. Eskimo
Samoyed


Toy dogs
Ancient & Spitz breeds
Spaniels
Scent hounds
Working dogs
Mastiff-like dogs
Small terriers
Retrievers
Herding dogs
Sight hounds
Wolves
Pomeranian
Chihuahua
Pekingese
Shih-Tzu
Brussels Griffon
Pug
Papillon
Min. Pin.
American Cocker Sp.
Eng. Cocker Sp.
Eng. Springer Sp.
Cavalier King Charles Sp.
Irish Water Sp.
Brittany Sp.
German Short-haired Pt.
Basset Hound
Beagle
Bloodhound
PBGV
Dachshund
Havanese
Std. Poodle
Toy Poodle
Dob. Pin.
Gt. Schnauzer
Std. Schnauzer
German Shep. Dog
Portuguese Water Dog
Boston Terr.
Boxer
Bulldog
French bulldog
Mins. Bull terr.
Std. Bull terr.
Glen of Imaal
Bull mastiff
Mastiff
Jack Russell
Aust. terr.
Briard
Wheat. Terr.
Cairn terr.
Highland terr.
Norwich terr.
Scottish terr.
Bernese Mtn. Dog
Std. Bernard
Grt. Dane
Rottweiler
Flat-coated Ret.
Golden Ret.
Labrador Ret.
Newfoundland
Aust. Shep.
Collie
Shetland Sheep dog
Border Collie
Cardigan Corgi
Pembroke Corgi
Old Eng. Sheep Dog
Bearded Collie
Scottish Deerhound
Irish Wolfhound
Greyhound
Whippet
It. Greyhound
Ibizan Hound
Kuvasz
Coyote
China
Balkans, Eastern & Northern Europe
Italy
Spain
Middle East
Near East
Basenji
Akita
Chow-chow
Shar-pei
Dingo
Alaskan Malamute
Sib. Husky
Afghan Hound
Saluki
Am. Eskimo
Samoyed


Plantae
Protista
Animalia
Cormophyta
Anthophyta
Angiospermae
Gymnospermae
Pteridophyta
Lepidophyta
Rhizocarpeae
Filices
Calamophyta
Bryophyta
Phyllobrya
Thallobrya
Fucoideae
Sargassaceae
Laminariaceae
Chordariaceae
Florideae
Sphaerococcaceae
Ceramiaceae
Characeae
Jnophyta
Lichenes
Fungi
Archephyta
Ulva
Conferva
Desmidium
Nostoc
Codiolum
Myxomycetes
Physarum
Stemonitis
Lycogala
Trichia
Spongiae
Petrospongiae
Siphonida
Ocellarida
Lymnorida
Bothrocnida
Turonida
Autospongiae
Calcispongiae
Silicispongiae
Ceratospongiae
Myxospongiae
Rhizopoda
Radiolaria
Actinophryida
Acyttaria
Polythalamia
Flagellata
Peridinium
Euglena
Volvox
Myxocytoda
Noctiluca
Diatomeae
Areolatae
Vittatae
Striatae
Protoplasta
Arcellae
Gregarinae
Autamoebae
Amoebae
Moneres
Protogenes
Protamoeba
Vampyrella
Protomonas
Vibrio
Vertebrata
Amniota
Aves
Mammalia
Reptilia
Anamnia
Amphibia
Pisces
Amphibina
Monorrhina
Leptocardia
Articulata
Arthropoda
Tracheata
Crustacea
Vermes
Annelida
Rotatoria
Scolecida
Infusoria
Echinodermata
Holothuriae
Echinida
Crinoida
Asterida
Mollusca
Otocardia
Himatega
Coelenterata
Nectacalephae
Petracalephae
Archephylum vegetabile
Archephylum protisticum
Archephylum animale
I, Feld: p m n q (19 Stämme)
II, Feld: p x y q (3 Stämme)
III, Feld: p s t q (1 Stamm)
stellen 3 mögliche Fälle der universalen Genealogie dar
Radix communis Organismorum
Moneres autogonum
Monophyletischer Stammbaum der Organismen
entworfen und gezeichnet von
Ernst Haeckel. Jena, 1866.


Ancient & Spitz breeds
Wolves
Balkans, Eastern & Northern Europe
China
Coyote
Kuvasz
Ibizan Hound
Sight hounds
Greyhound
Herding dogs
Retrievers


Animal
Plant
Bacteria
Fungi
Archaea
Protista etc.
Animalia
Homo sapiens
Fungi
Archaea
Life on Earth
Bacteria
Protista etc.
Plantae
The Tree of Life
This drawing illustrates the phylogeny of organisms, i.e., the history of organismal lineages as they change through time, as documented by the Tree of Life Web Project. Each dot represents a species or a group, and is linked to its containing group by an edge. Data collected from the project website (http://www.tolweb.org) on 9/19/2011. A total of 93891 species and groups were found on the site, a small sample of the estimated millions of species on Earth today.

Ancient &

Wolv

Sight hounds

The Tree of Life

This drawing illustrates the phylogeny of organisms, i.e., the history of organismal lineages as they change through time, as documented by the Tree of Life Web Project. Each dot represents a species or a group, and is linked to its containing group by an edge. Data collected from the project website (http://www.tolweb.org) on 9/19/2011. A total of 93891 species and groups were found on the site, a small sample of the estimated millions of species on Earth today.

Illustration created by Yifan Hu, AT&T Labs – Research

Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees

Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees
- different data (e.g., due to measuring errors) yield different trees

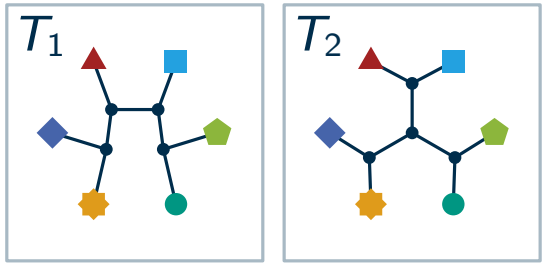
Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees
- different data (e.g., due to measuring errors) yield different trees
- How to compare different trees T_1 and T_2 on the same set of leaves L?



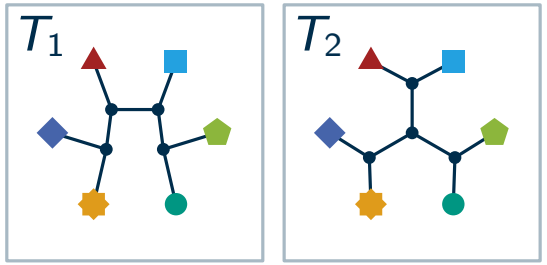
Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees
- different data (e.g., due to measuring errors) yield different trees
- How to compare different trees T_1 and T_2 on the same set of leaves L?



Maximum agreement forest

- forest F of full binary trees with leaves L
- trees in F are contained in T_1 and T_2 (as minors)

Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

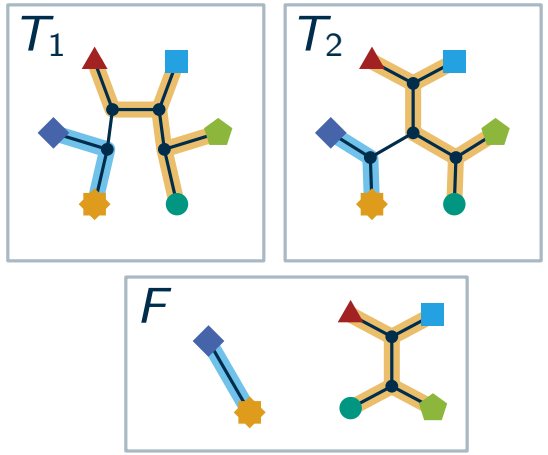
$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees
- different data (e.g., due to measuring errors) yield different trees
- How to compare different trees T_1 and T_2 on the same set of leaves L?

Maximum agreement forest

- forest F of full binary trees with leaves L
- trees in F are contained in T_1 and T_2 (as minors)



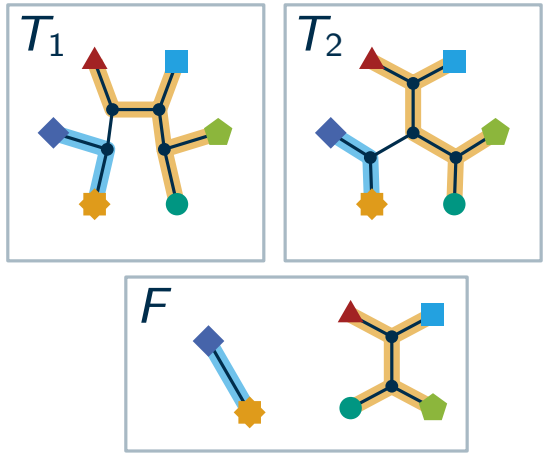
Phylogenetic trees

Definition
A **phylogenetic tree** is an unrooted full binary tree, such that each leaf has a unique label.

$\rightarrow$ are often generated automatically, e.g., based on DNA sequencing

Problem

- different algorithms yield different trees
- different data (e.g., due to measuring errors) yield different trees
- How to compare different trees T_1 and T_2 on the same set of leaves L?



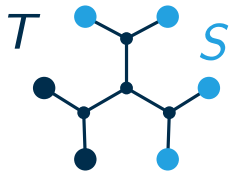
Maximum agreement forest

- forest F of full binary trees with leaves L
- trees in F are contained in T_1 and T_2 (as minors)
- minimize the number of trees in F

What does this mean exactly?

Leaf-induced subtrees

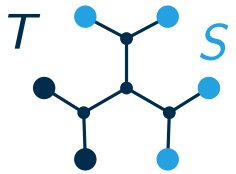
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$

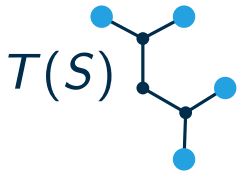


What does this mean exactly?

Leaf-induced subtrees

- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S) =$ minimal subtree of T that contains S

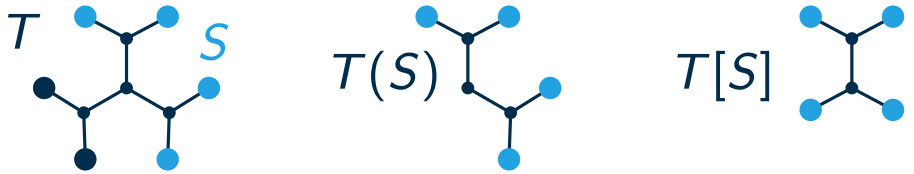




What does this mean exactly?

Leaf-induced subtrees

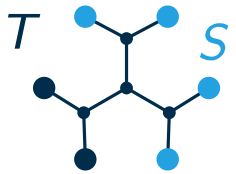
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S) =$ minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

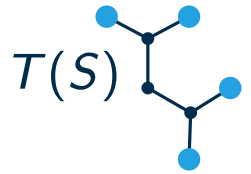


What does this mean exactly?

Leaf-induced subtrees

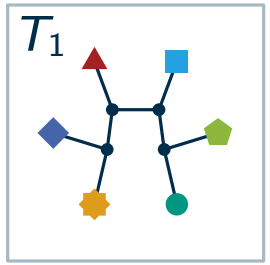
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S)$ = minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

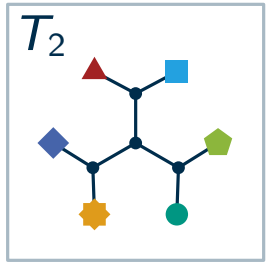


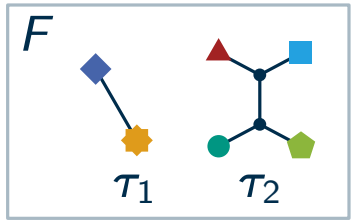


Agreement forest

- let F be a forest consisting of the trees $\tau_1, \ldots, \tau_k$
- T_1 and T_2 are trees (same leaves $L(T_1) = L(T_2) = L(F)$)



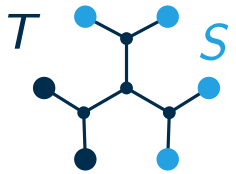


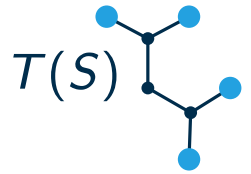


What does this mean exactly?

Leaf-induced subtrees

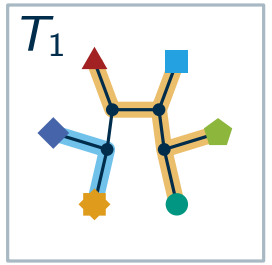
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S)$ = minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

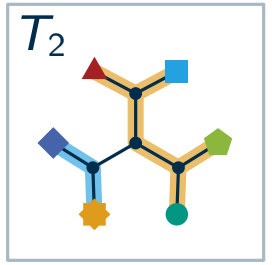


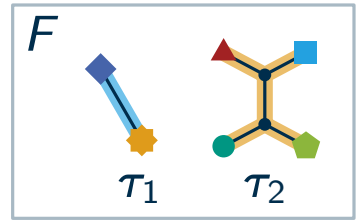


Agreement forest

- let F be a forest consisting of the trees $\tau_1, \ldots, \tau_k$
- T_1 and T_2 are trees (same leaves $L(T_1) = L(T_2) = L(F)$)
- F is an **agreement forest** of T_1 and T_2, if:
  - each τ_i included in T_1 and T_2: $T_1[L(\tau_i)] = T_2[L(\tau_i)] = \tau_i$
  - disjoint τs: $T(L(\tau_i)) \cap T(L(\tau_j)) = \emptyset$ (for $T \in \{T_1, T_2\}$)



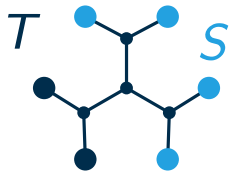


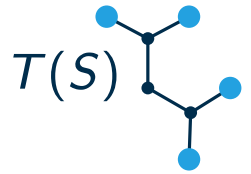


What does this mean exactly?

Leaf-induced subtrees

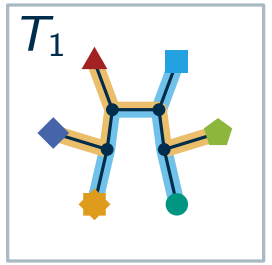
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S)$ = minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

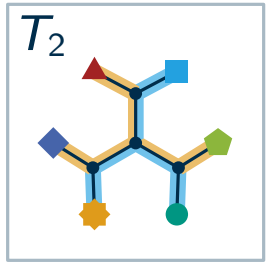


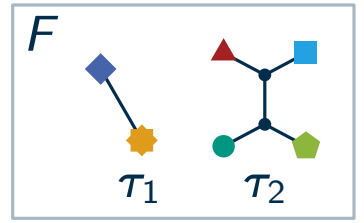


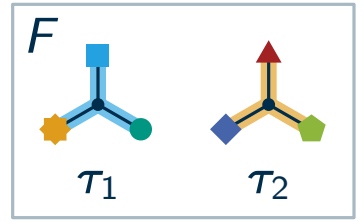
Agreement forest

- let F be a forest consisting of the trees $\tau_1, \ldots, \tau_k$
- T_1 and T_2 are trees (same leaves $L(T_1) = L(T_2) = L(F)$)
- F is an **agreement forest** of T_1 and T_2, if:
  - each τ_i included in T_1 and T_2: $T_1[L(\tau_i)] = T_2[L(\tau_i)] = \tau_i$
  - disjoint τs: $T(L(\tau_i)) \cap T(L(\tau_j)) = \emptyset$ (for $T \in \{T_1, T_2\}$)





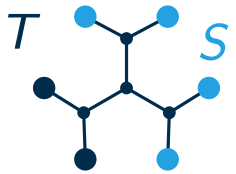


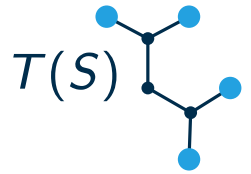


What does this mean exactly?

Leaf-induced subtrees

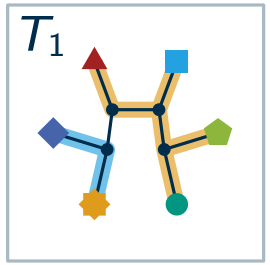
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S)$ = minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

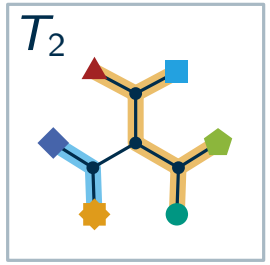


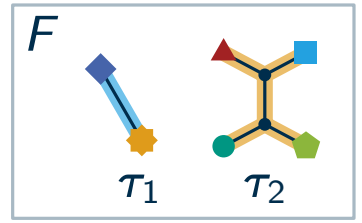


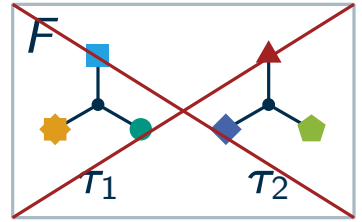
Agreement forest

- let F be a forest consisting of the trees $\tau_1, \ldots, \tau_k$
- T_1 and T_2 are trees (same leaves $L(T_1) = L(T_2) = L(F)$)
- F is an **agreement forest** of T_1 and T_2, if:
  - each τ_i included in T_1 and T_2: $T_1[L(\tau_i)] = T_2[L(\tau_i)] = \tau_i$
  - disjoint τs: $T(L(\tau_i)) \cap T(L(\tau_j)) = \emptyset$ (for $T \in \{T_1, T_2\}$)







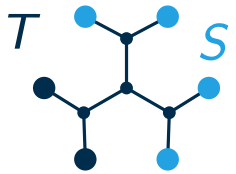


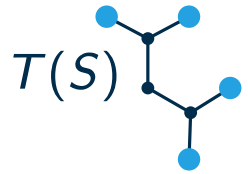
(violates disjoint τs)

What does this mean exactly?

Leaf-induced subtrees

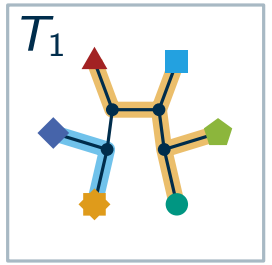
- let T be a tree with leaves $L(T)$ and let $S \subseteq L(T)$
- $T(S) =$ minimal subtree of T that contains S
- contract all vertices of degree 2 in $T(S) \to T[S]$

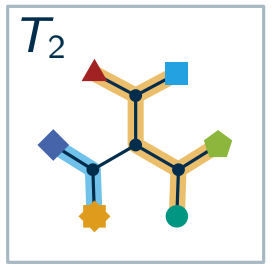


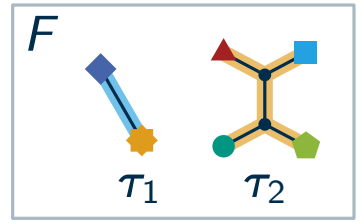


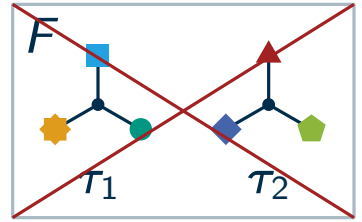
Agreement forest

- let F be a forest consisting of the trees $\tau_1, \ldots, \tau_k$
- T_1 and T_2 are trees (same leaves $L(T_1) = L(T_2) = L(F)$)
- F is an **agreement forest** of T_1 and T_2, if:
  - each τ_i included in T_1 and T_2: $T_1[L(\tau_i)] = T_2[L(\tau_i)] = \tau_i$
  - disjoint τs: $T(L(\tau_i)) \cap T(L(\tau_j)) = \emptyset$ (for $T \in \{T_1, T_2\}$)









(violates disjoint τs)

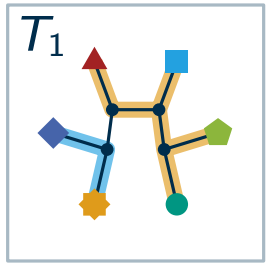
Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

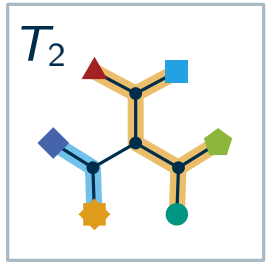
Plan: Kernelization

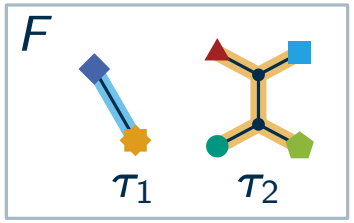
Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Goal

- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k







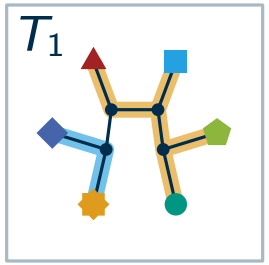
Plan: Kernelization

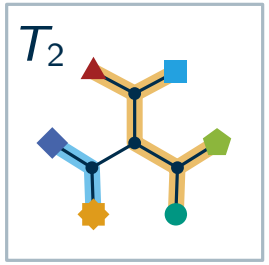
Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

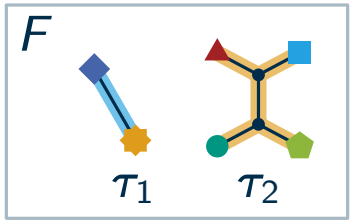
Goal

- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

Suggestions?







Plan: Kernelization

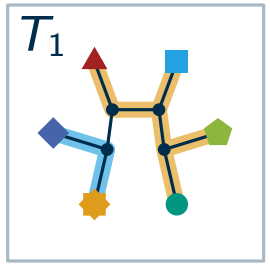
Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

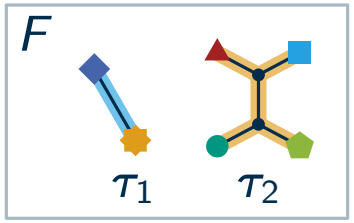
Goal

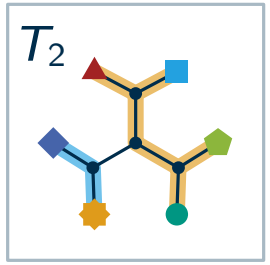
- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

Suggestions?

Before we start reducing: What is the goal?







Plan: Kernelization

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

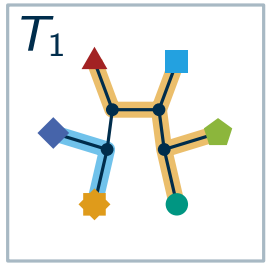
Goal

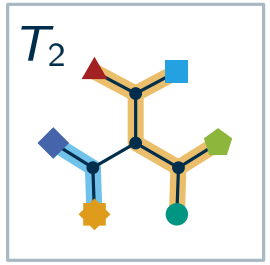
- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

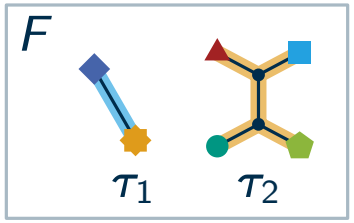
Suggestions?

Before we start reducing: What is the goal?

- size of the resulting trees only depends on k







Plan: Kernelization

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

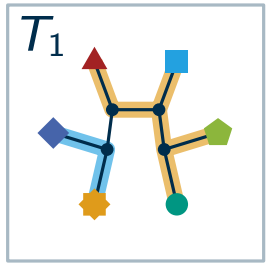
Goal

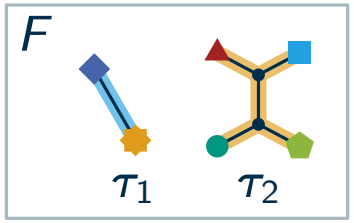
- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

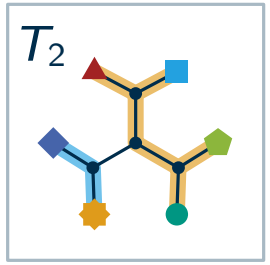
Suggestions?

Before we start reducing: What is the goal?

- size of the resulting trees only depends on k
- slightly stronger but more specific claim:







Plan: Kernelization

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

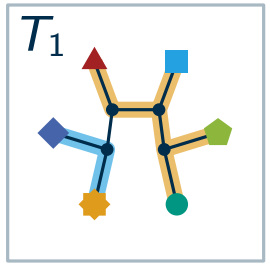
Goal

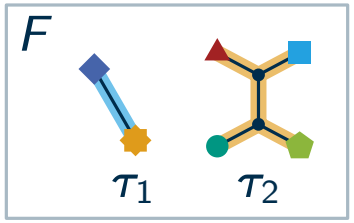
- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

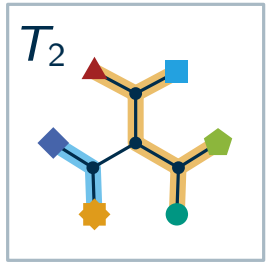
Suggestions?

Before we start reducing: What is the goal?

- size of the resulting trees only depends on k
- slightly stronger but more specific claim:
  - every tree in the agreement forest has few leaves







Plan: Kernelization

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

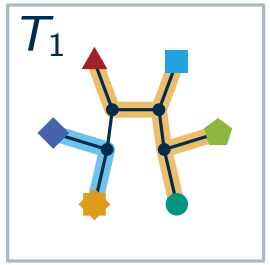
Goal

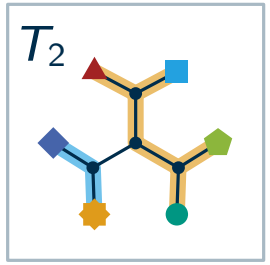
- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

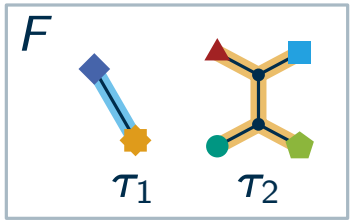
Suggestions?

Before we start reducing: What is the goal?

- size of the resulting trees only depends on k
- slightly stronger but more specific claim:
  - every tree in the agreement forest has few leaves
  - implies: few leaves overall $\Rightarrow$ small instance







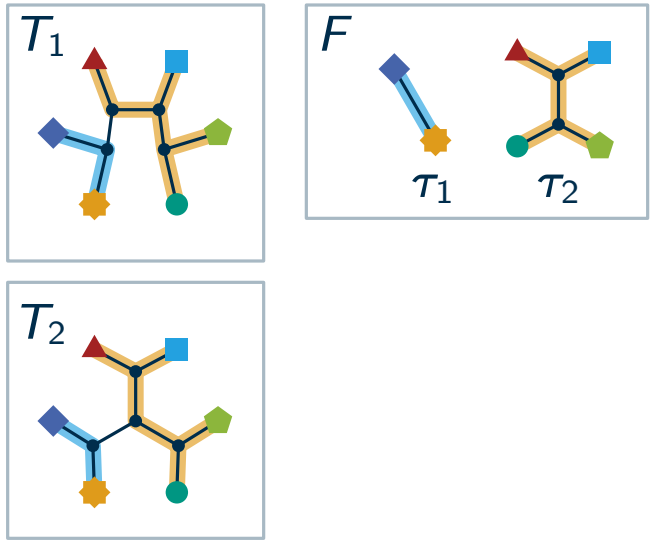
Plan: Kernelization

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Goal

- find a kernelization algorithm $\rightarrow$ polynomial reduction rules
- size of the resulting trees only depends on k

Suggestions?



Before we start reducing: What is the goal?

- size of the resulting trees only depends on k
- slightly stronger but more specific claim:
  - every tree in the agreement forest has few leaves
  - implies: few leaves overall $\Rightarrow$ small instance
- Which tree structures contradict this claim? Can we remove them using reduction rules?

Undesirable tree structure (1)

Claim: Every tree of the agreement forest has only few leaves.

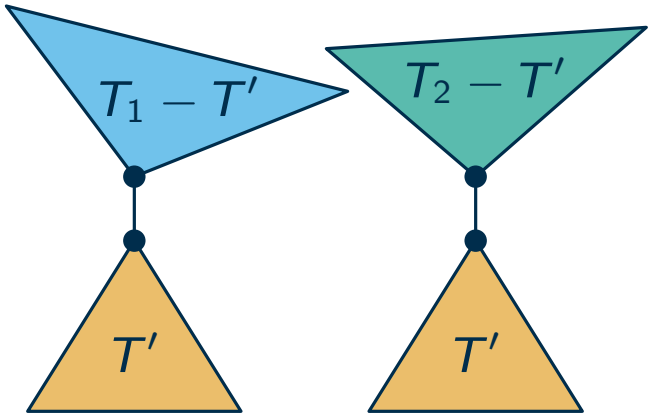
Counterexample

Undesirable tree structure (1)

Claim: Every tree of the agreement forest has only few leaves.

Counterexample

- T_1 and T_2 may share a subtree T' with many leaves
- in the agreement forest, T' could be one of the $\tau_1, \ldots, \tau_k$

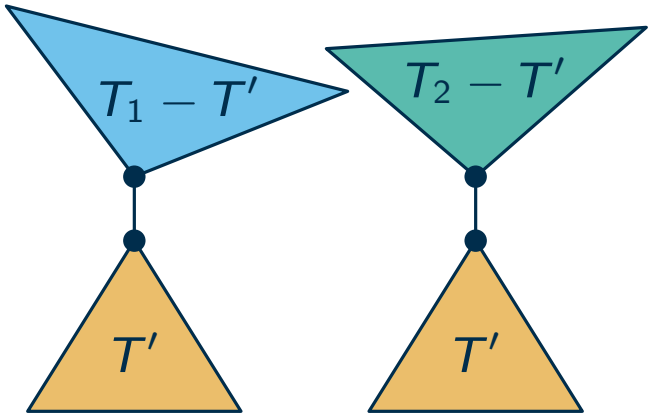


Undesirable tree structure (1)

Claim: Every tree of the agreement forest has only few leaves.

Counterexample

- T_1 and T_2 may share a subtree T' with many leaves
- in the agreement forest, T' could be one of the $\tau_1, \ldots, \tau_k$



Reduction rule 1

Undesirable tree structure (1)

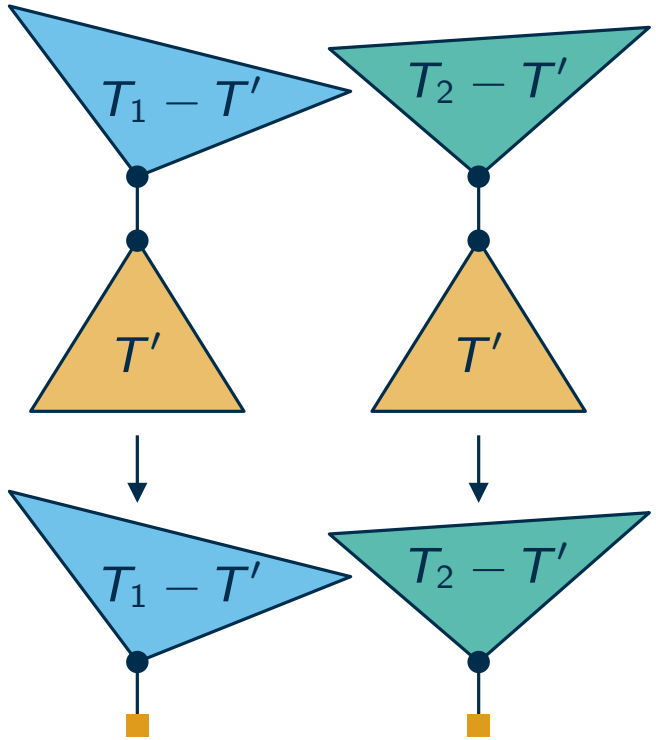
Claim: Every tree of the agreement forest has only few leaves.

Counterexample

- T_1 and T_2 may share a subtree T' with many leaves
- in the agreement forest, T' could be one of the $\tau_1, \ldots, \tau_k$

Reduction rule 1

- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)



Undesirable tree structure (1)

Claim: Every tree of the agreement forest has only few leaves.

Counterexample

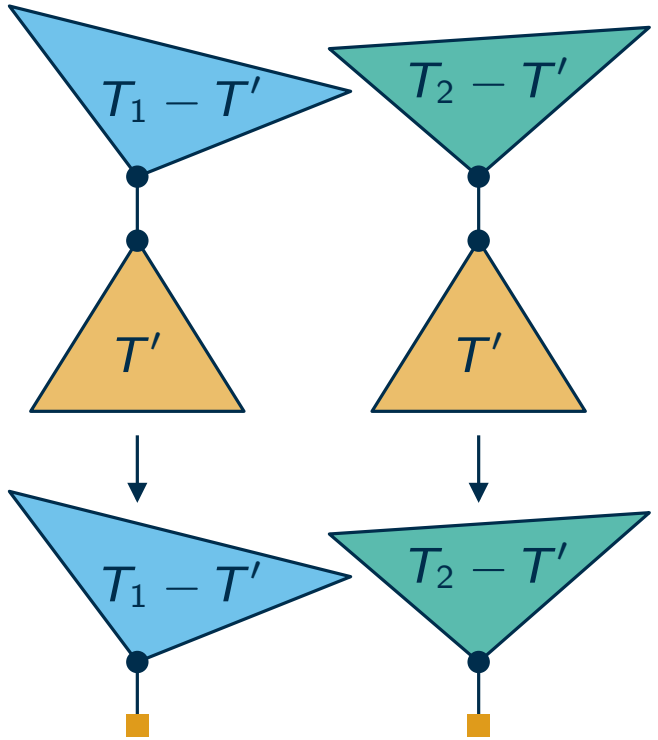
- T_1 and T_2 may share a subtree T' with many leaves
- in the agreement forest, T' could be one of the $\tau_1, \ldots, \tau_k$

Reduction rule 1

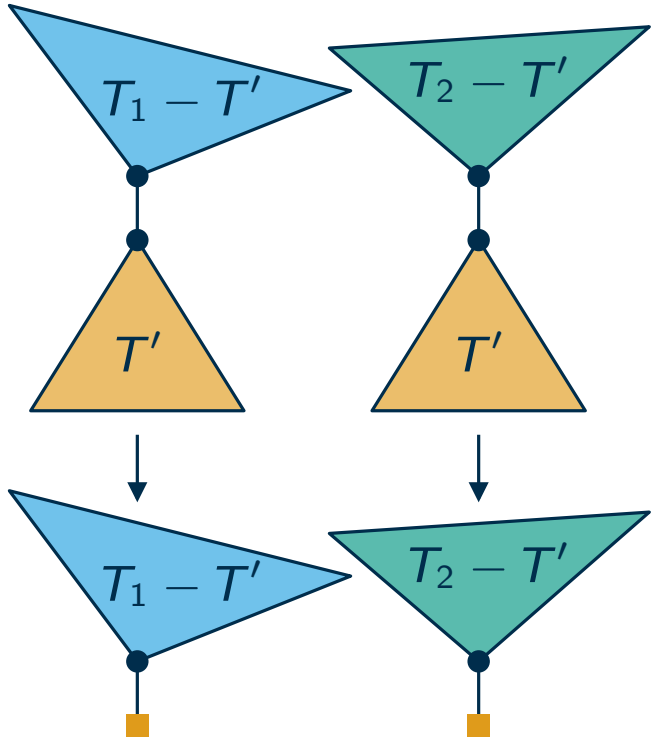
- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)

Lemma
Reduction rule 1 is safe and can be applied in polynomial time.

Proof: next slide



Reduction rule 1



Reduction rule 1

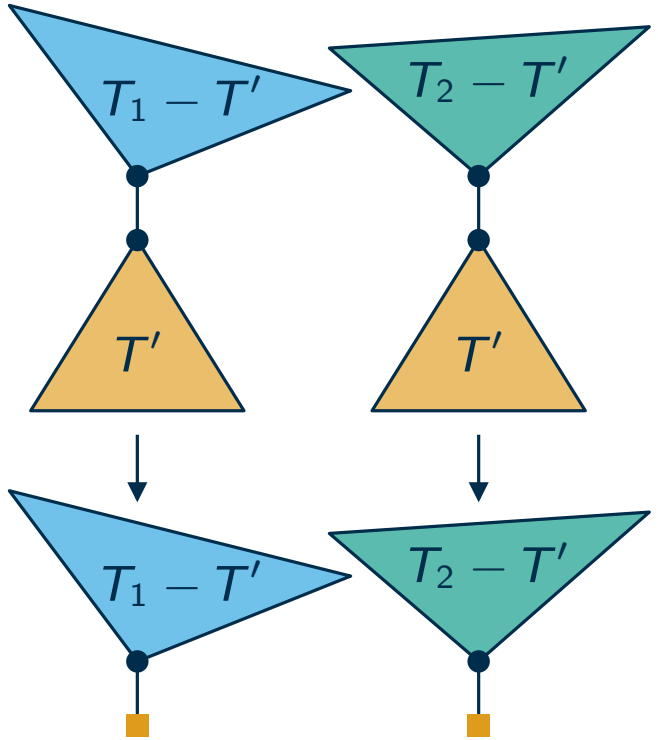
- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)

Lemma
Reduction rule 1 is safe and can be applied in polynomial time.

Proof

- let F' be an agreement forest of the new instance and let τ_1' be the tree with the new leaf in F'

Reduction rule 1



Reduction rule 1

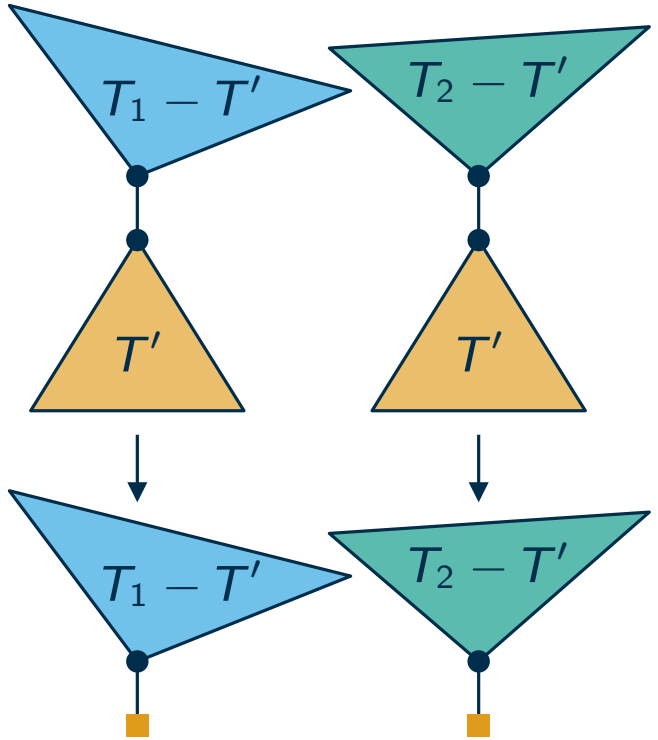
- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)

Lemma
Reduction rule 1 is safe and can be applied in polynomial time.

Proof

- let F' be an agreement forest of the new instance and let τ_1' be the tree with the new leaf in F'
- replace this leaf in τ_1' by $T' \rightarrow$ agreement forest of the same size for T_1 and T_2

Reduction rule 1



Reduction rule 1

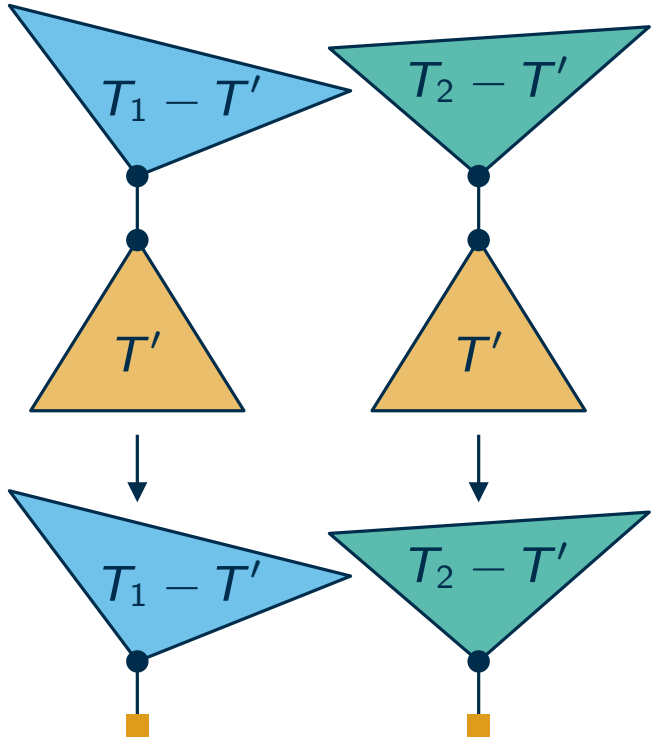
- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)

Lemma
Reduction rule 1 is safe and can be applied in polynomial time.

Proof

- let F' be an agreement forest of the new instance and let τ_1' be the tree with the new leaf in F'
- replace this leaf in τ_1' by $T' \rightarrow$ agreement forest of the same size for T_1 and T_2
- other direction: similar

Reduction rule 1



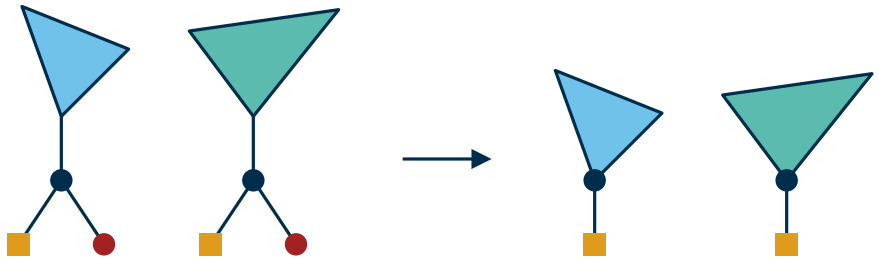
Reduction rule 1

- find edges in T_1 and T_2 that cut off the same subtree T'
- replace T' by a leaf with a new label (same label in T_1 and T_2)

Lemma
Reduction rule 1 is safe and can be applied in polynomial time.

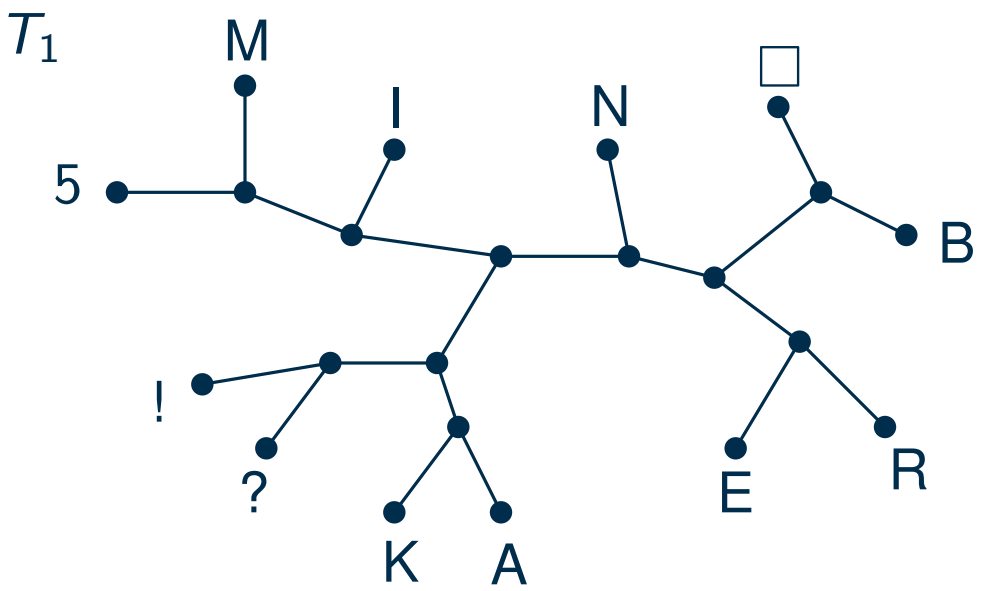
Proof

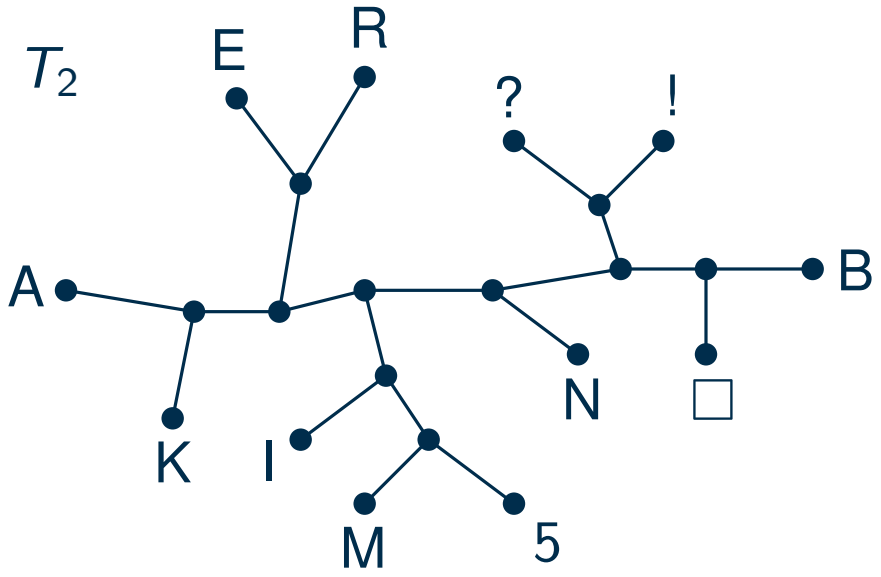
- let F' be an agreement forest of the new instance and let τ_1' be the tree with the new leaf in F'
- replace this leaf in τ_1' by $T' \rightarrow$ agreement forest of the same size for T_1 and T_2
- other direction: similar
- polynomial running time: for example by iteratively eating “cherries”



Example

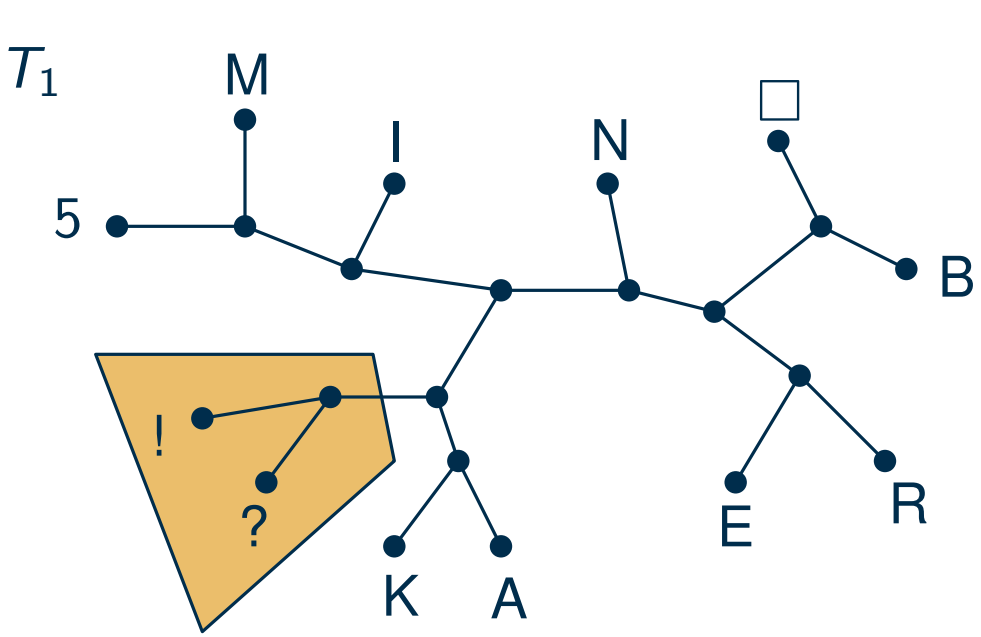
Solve this instance of MAXIMUM AGREEMENT FOREST

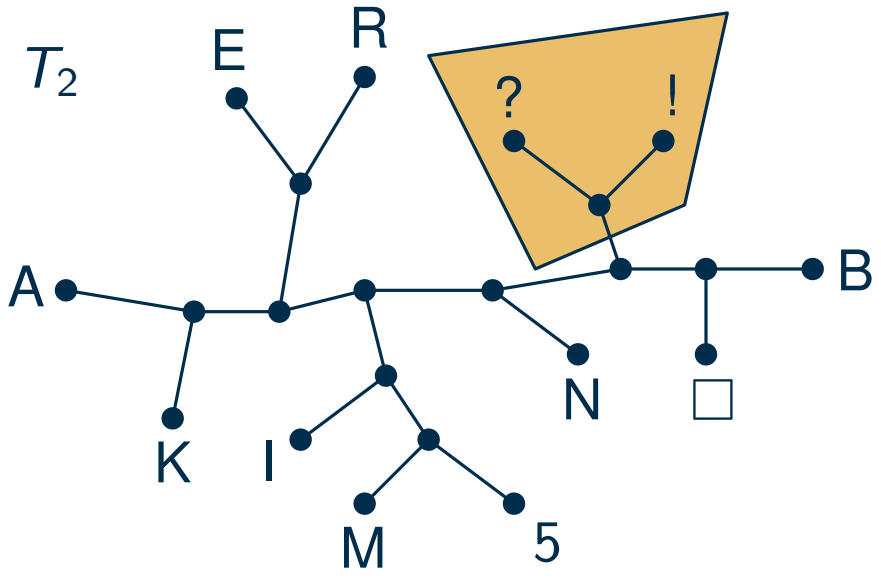




Example

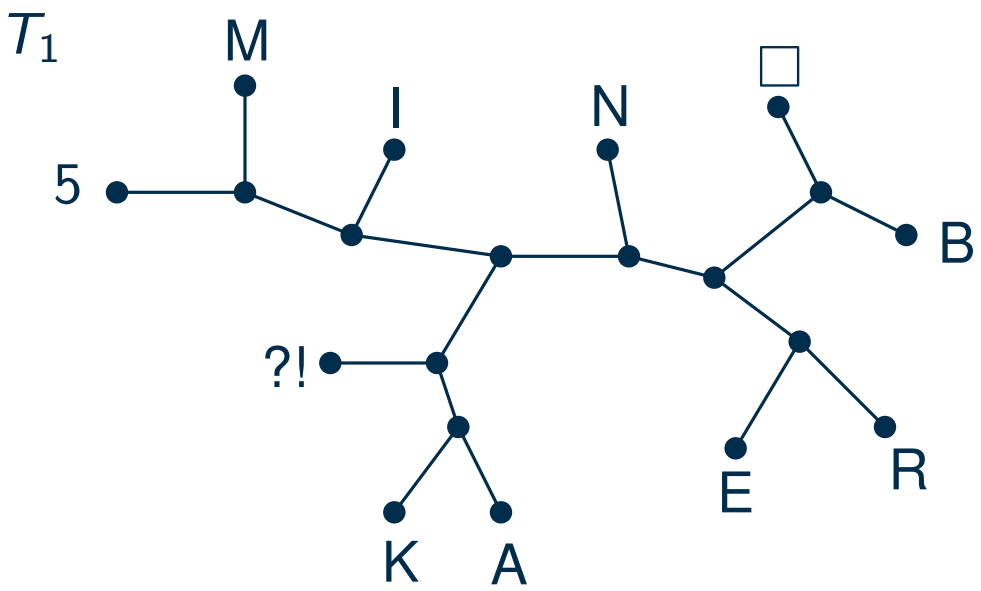
Solve this instance of MAXIMUM AGREEMENT FOREST

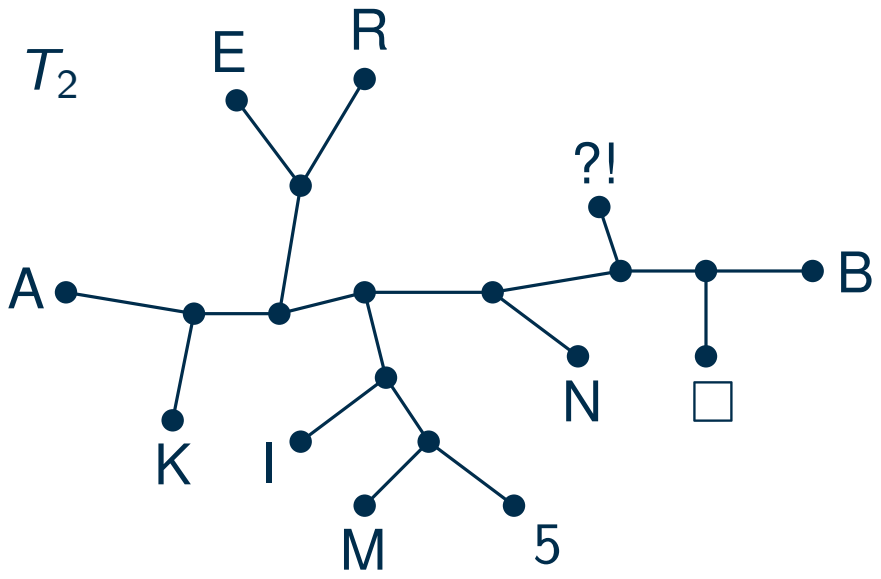




Example

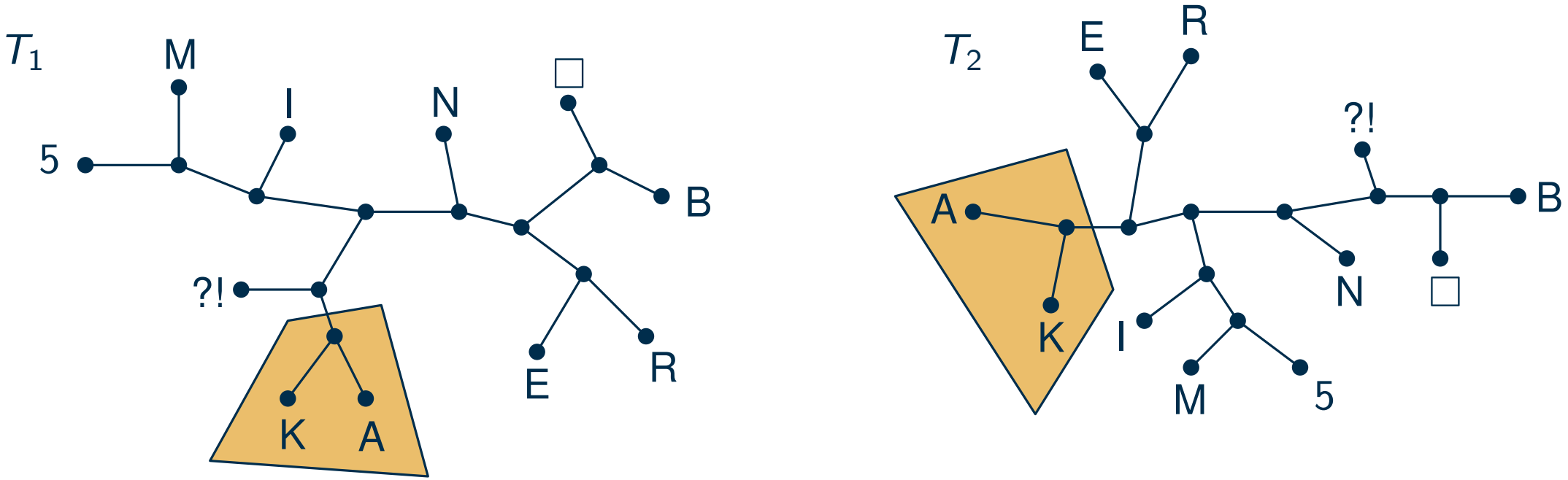
Solve this instance of MAXIMUM AGREEMENT FOREST





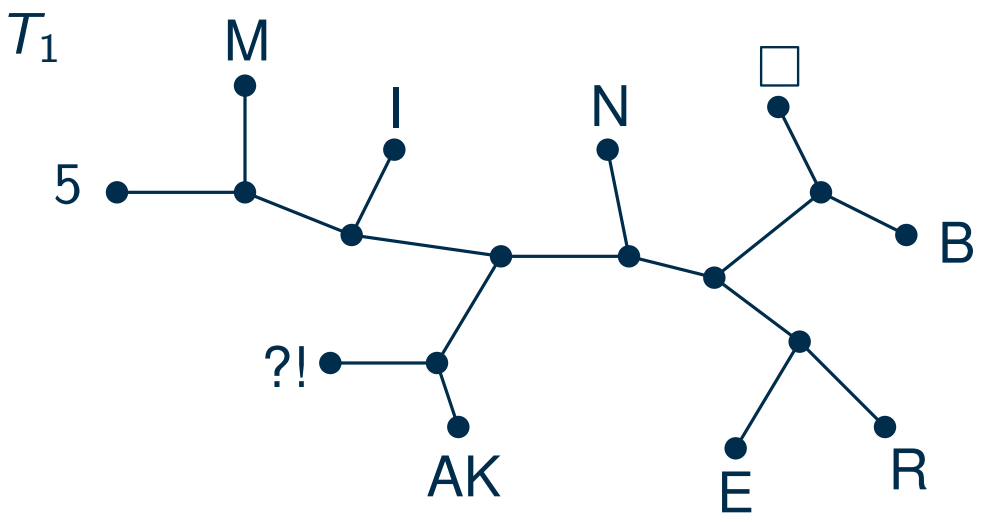
Example

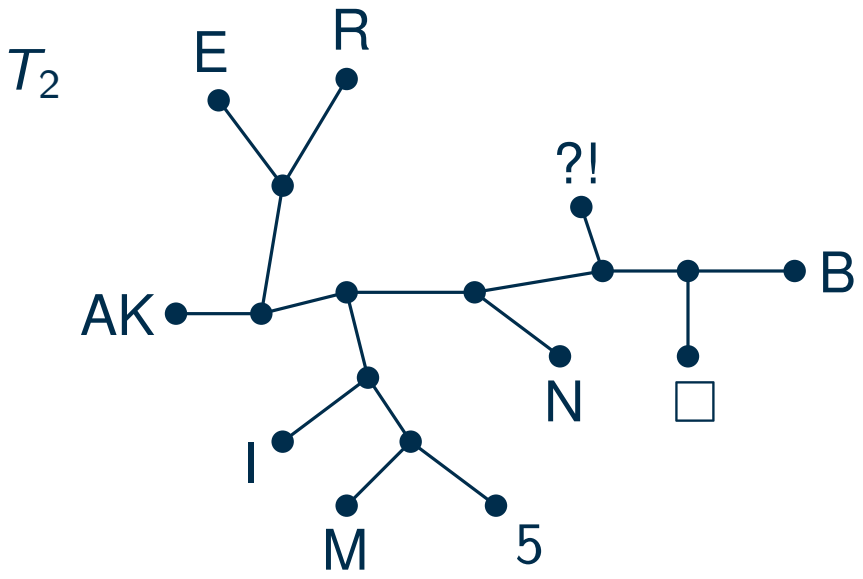
Solve this instance of MAXIMUM AGREEMENT FOREST



Example

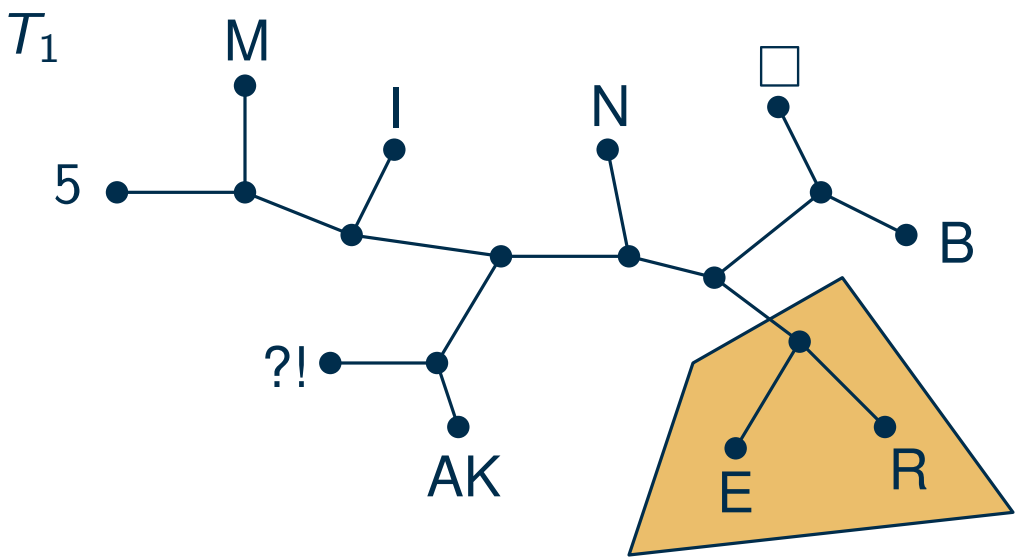
Solve this instance of MAXIMUM AGREEMENT FOREST

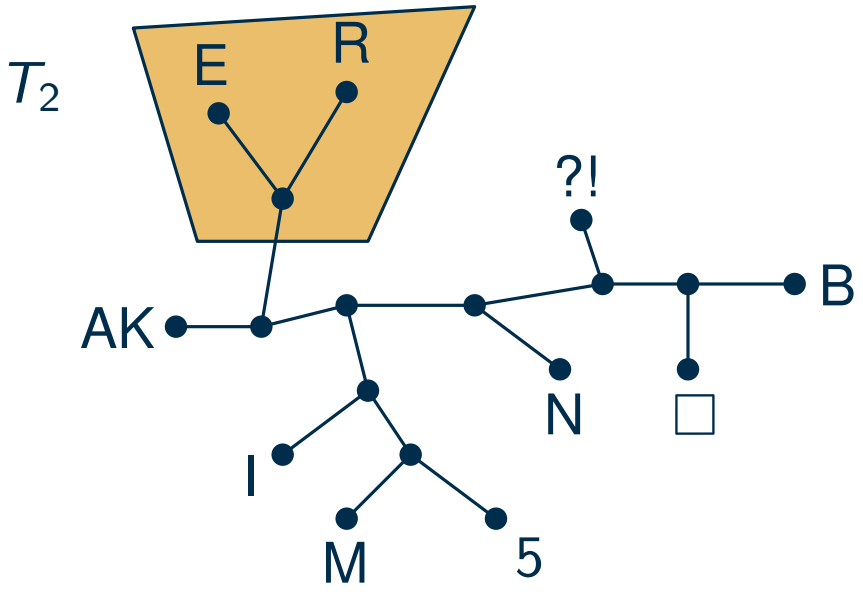




Example

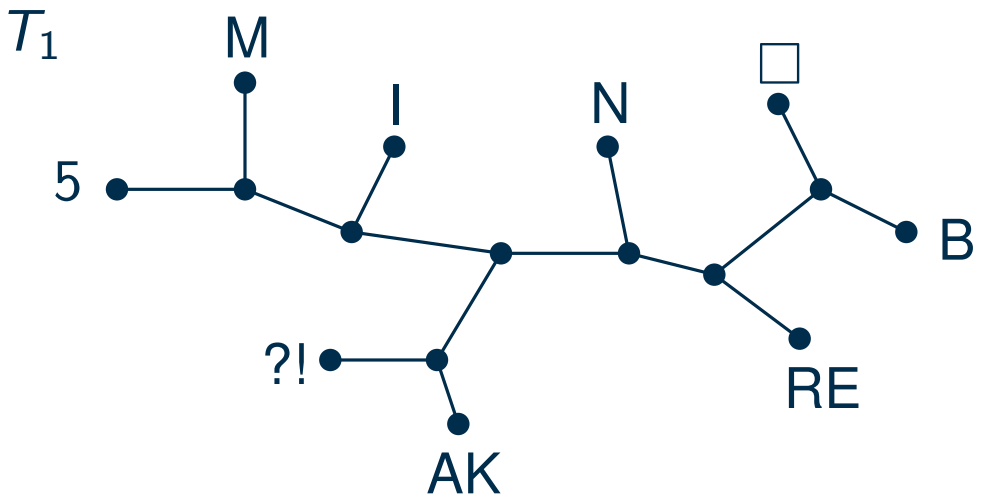
Solve this instance of MAXIMUM AGREEMENT FOREST

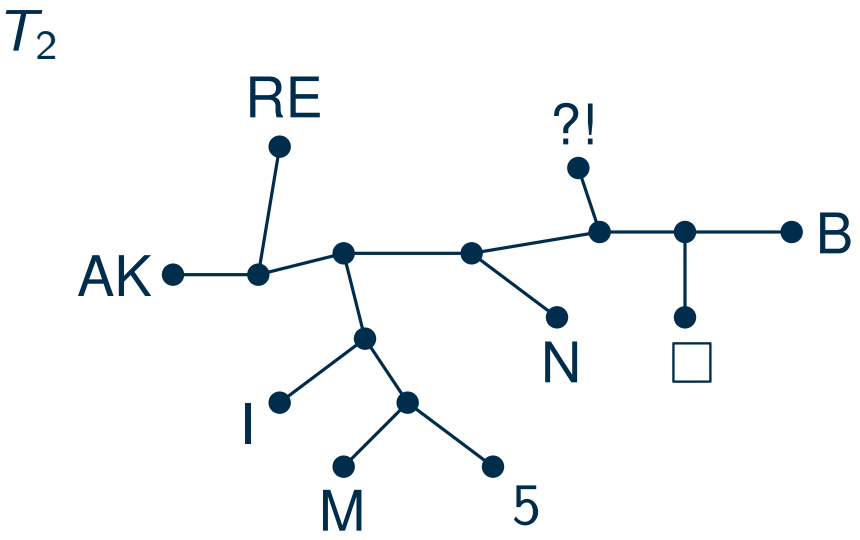




Example

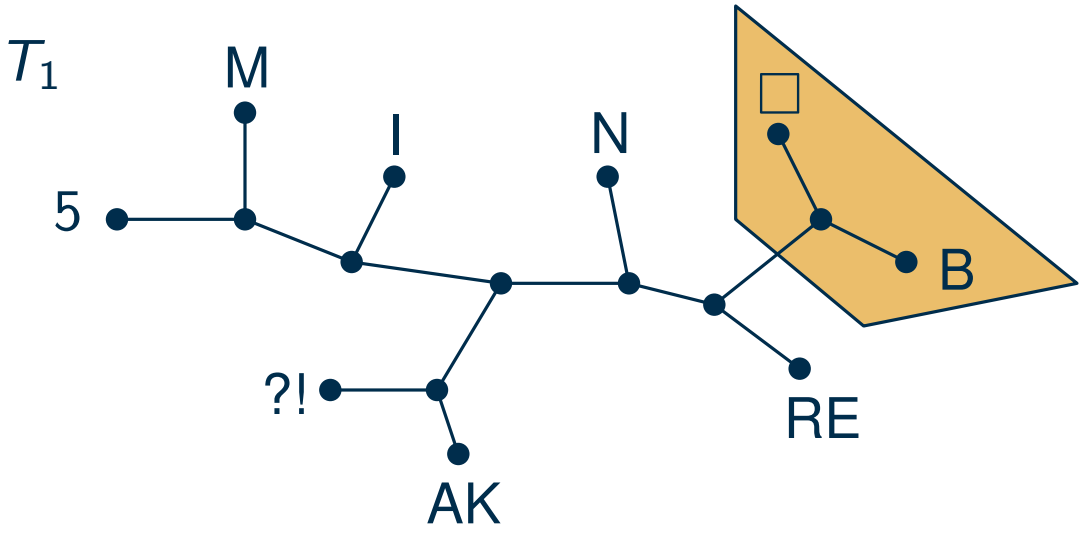
Solve this instance of MAXIMUM AGREEMENT FOREST

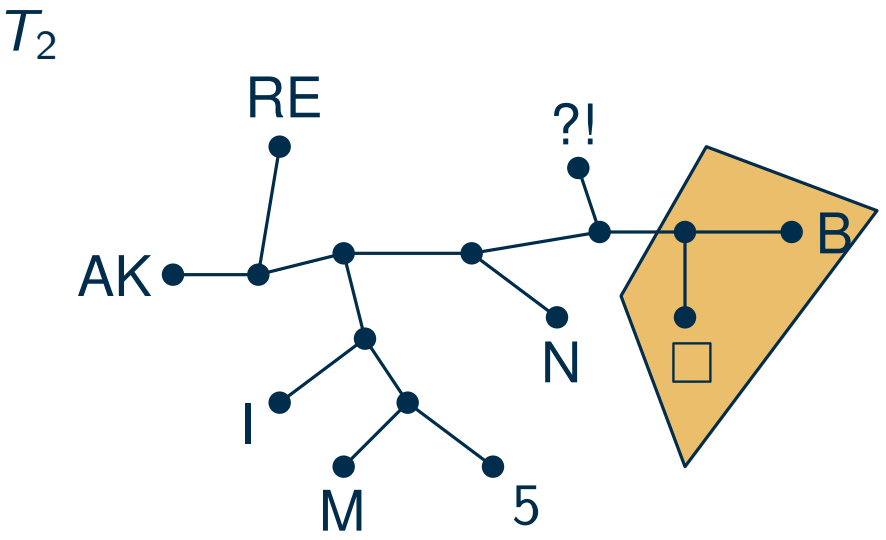




Example

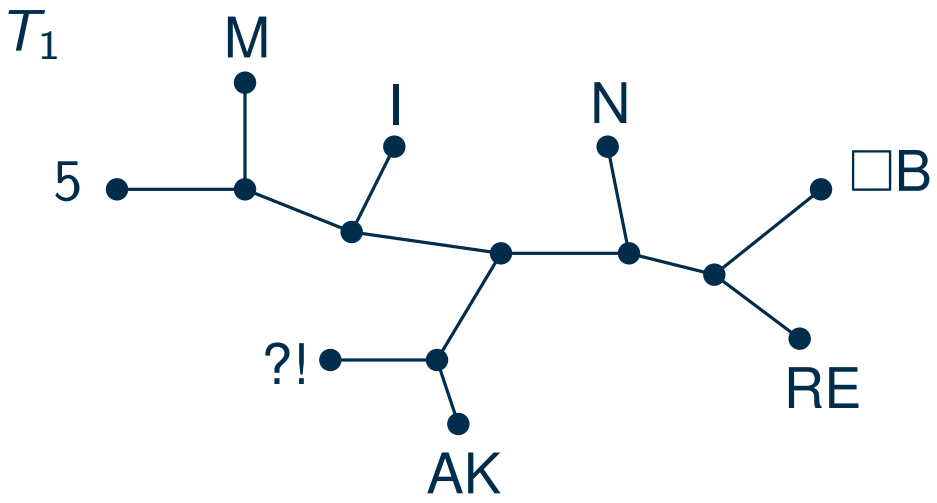
Solve this instance of MAXIMUM AGREEMENT FOREST

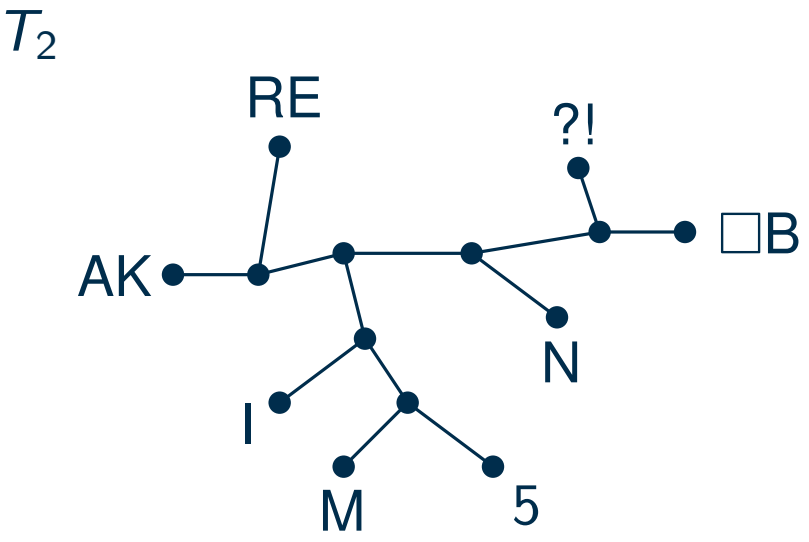




Example

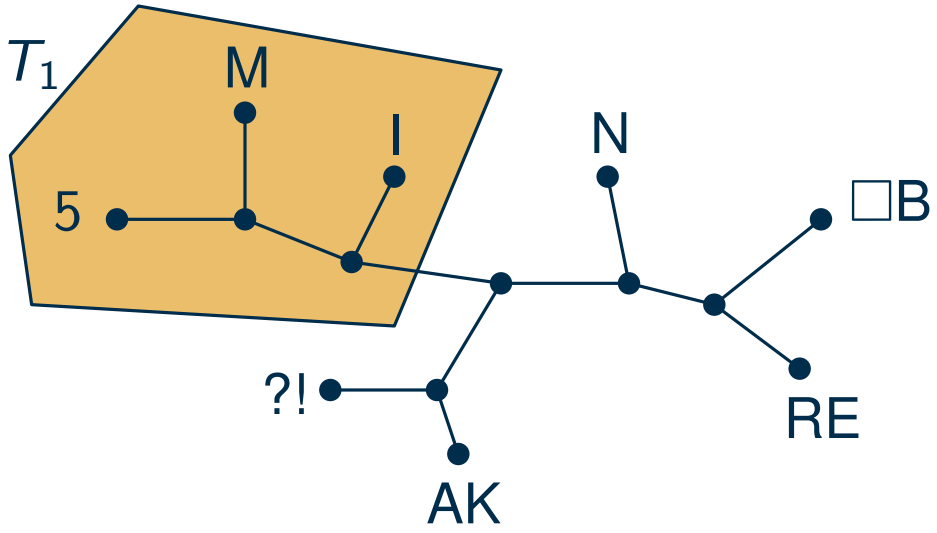
Solve this instance of MAXIMUM AGREEMENT FOREST

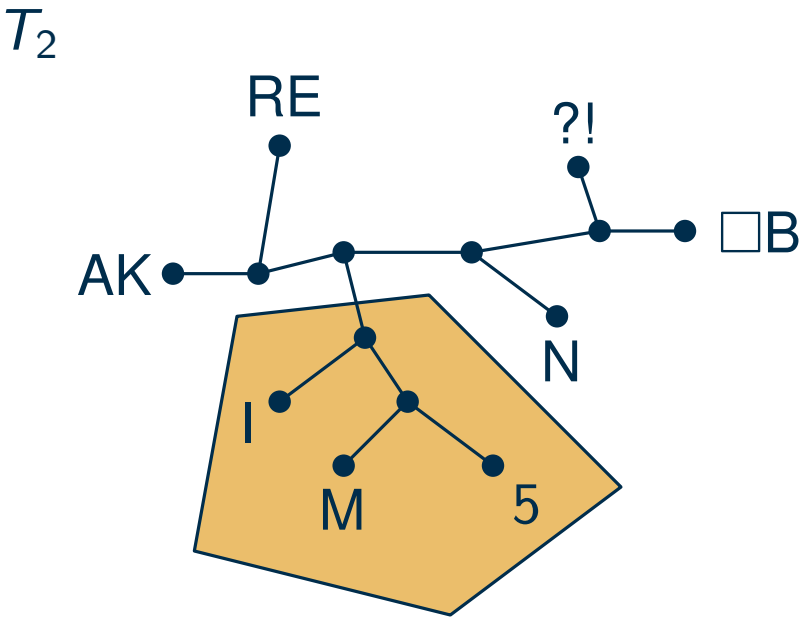




Example

Solve this instance of MAXIMUM AGREEMENT FOREST

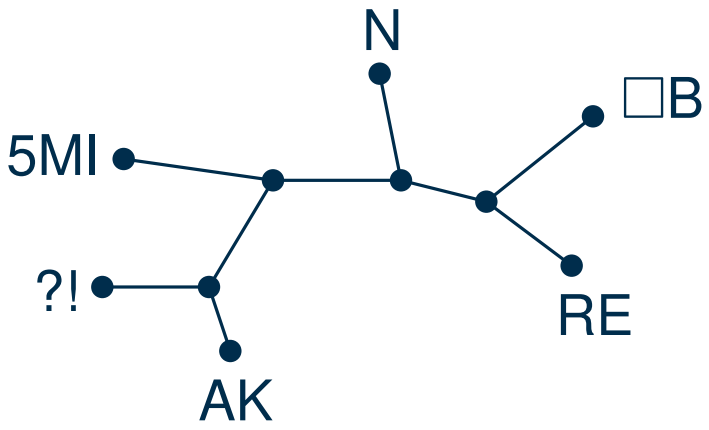




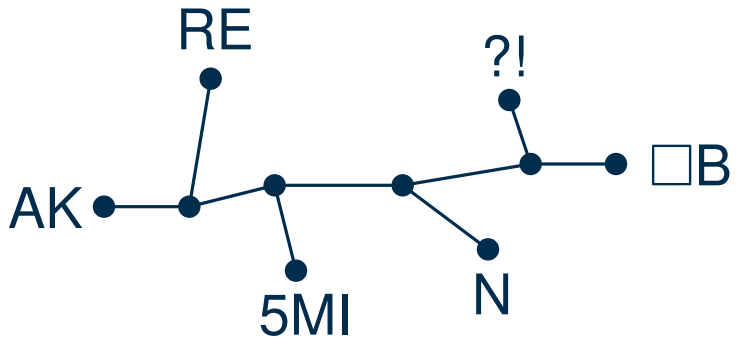
Example

Solve this instance of MAXIMUM AGREEMENT FOREST

T_1



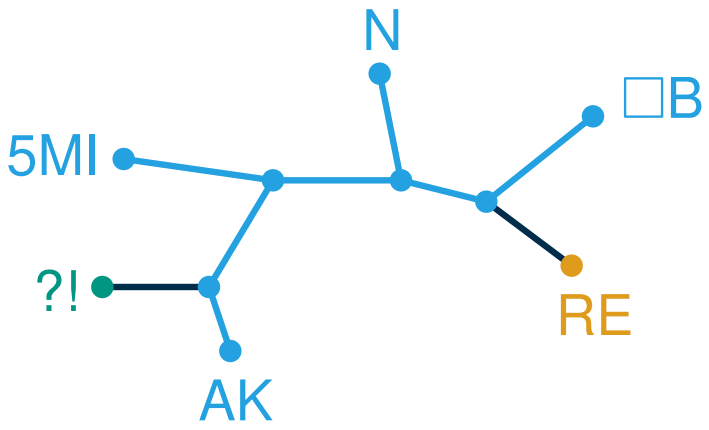
T_2



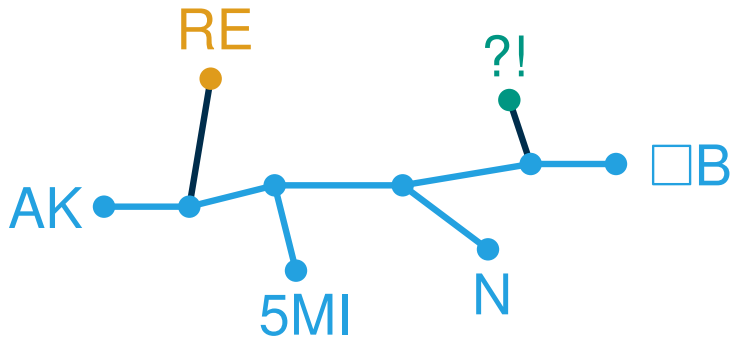
Example

Solve this instance of MAXIMUM AGREEMENT FOREST

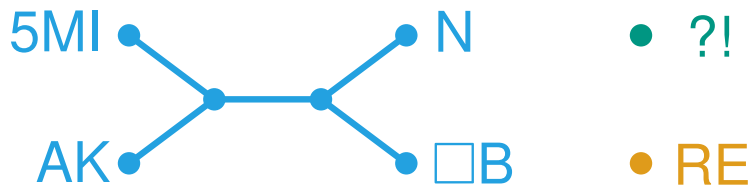
T_1



T_2



F



$\Rightarrow$ solution has size 3

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

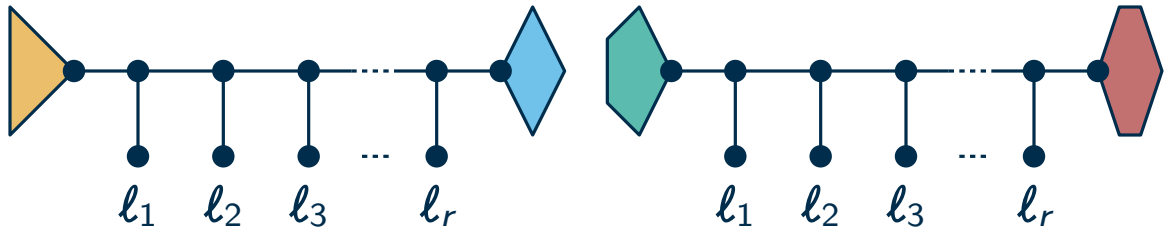
Counterexample

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree

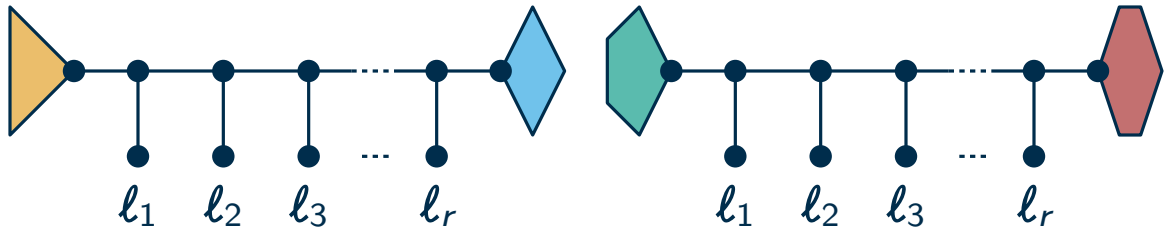


Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



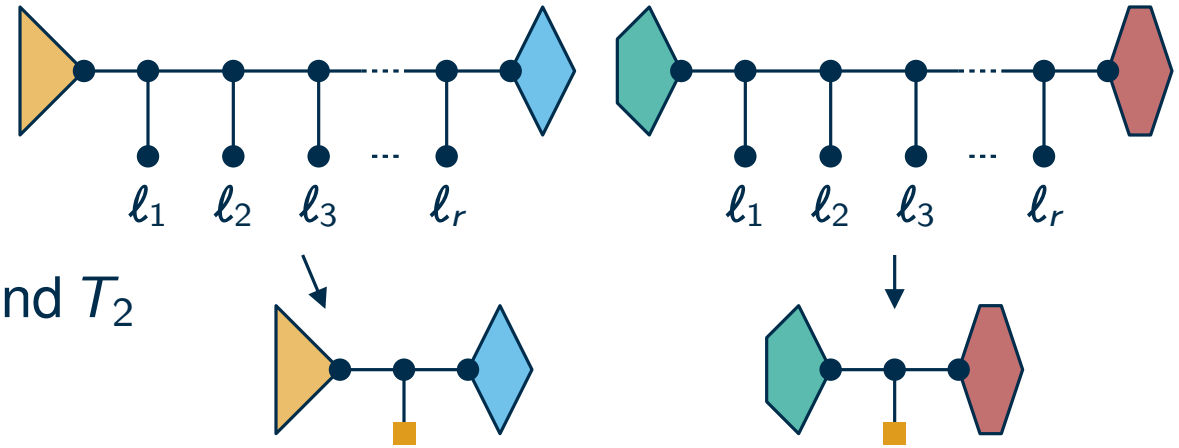
Reduction rule 2

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace the path by a vertex with a new leaf

Undesirable tree structures 2

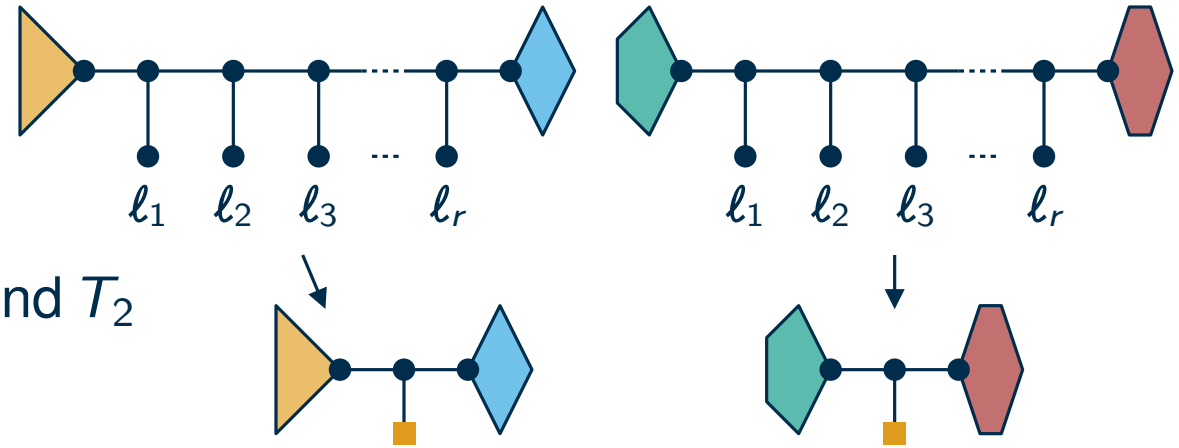
Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree

Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace the path by a vertex with a new leaf



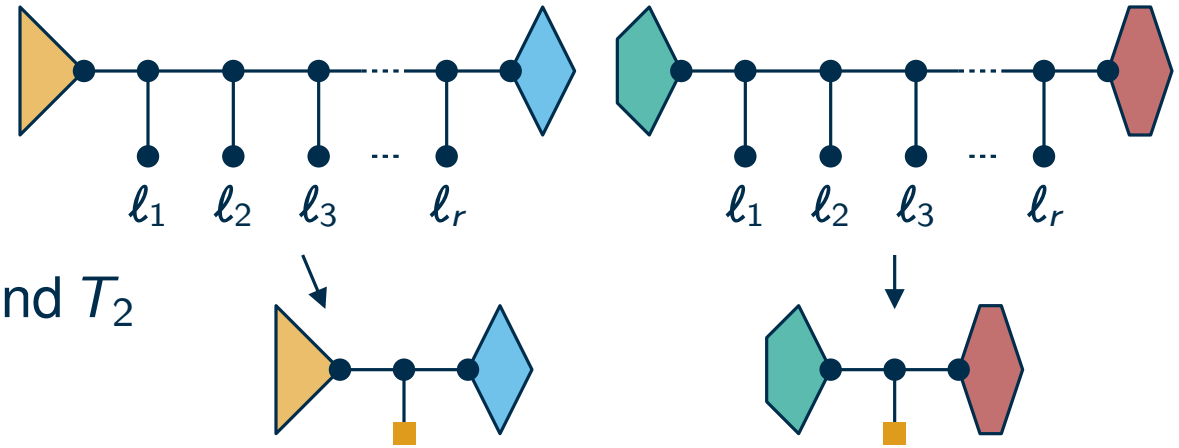
Is this rule safe?

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



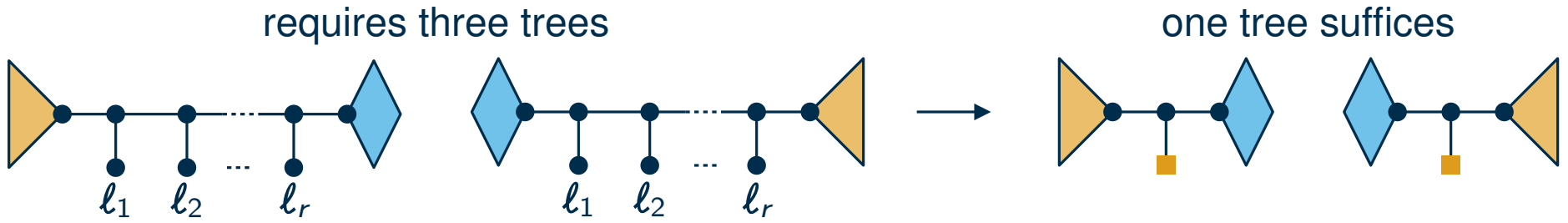
Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace the path by a vertex with a new leaf

Problem

- number of required trees may be reduced

Is this rule safe?

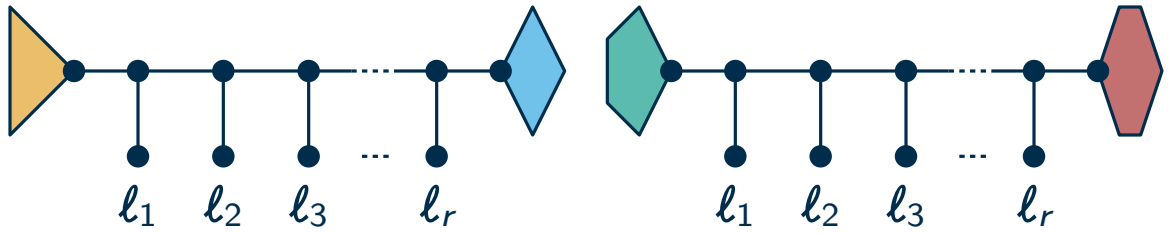


Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



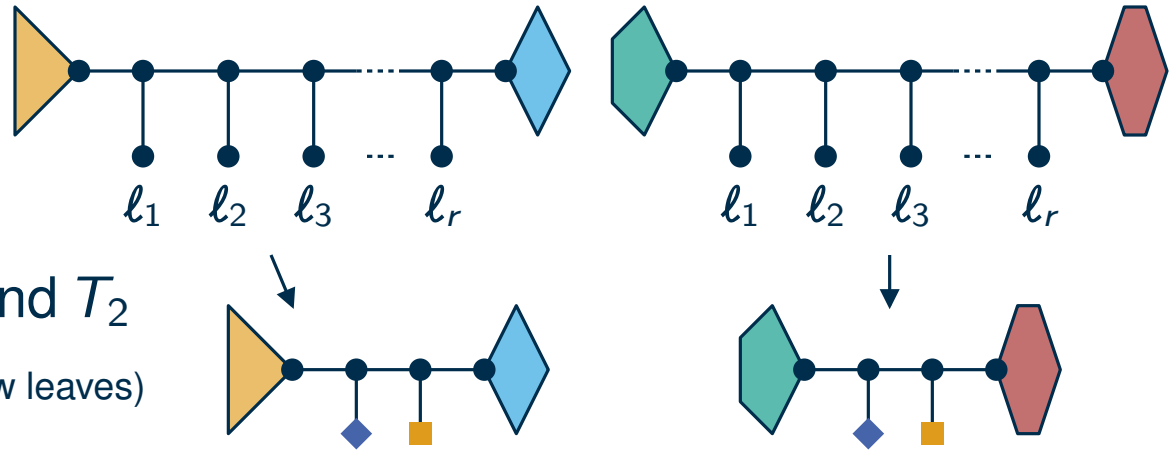
Reduction rule 2

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

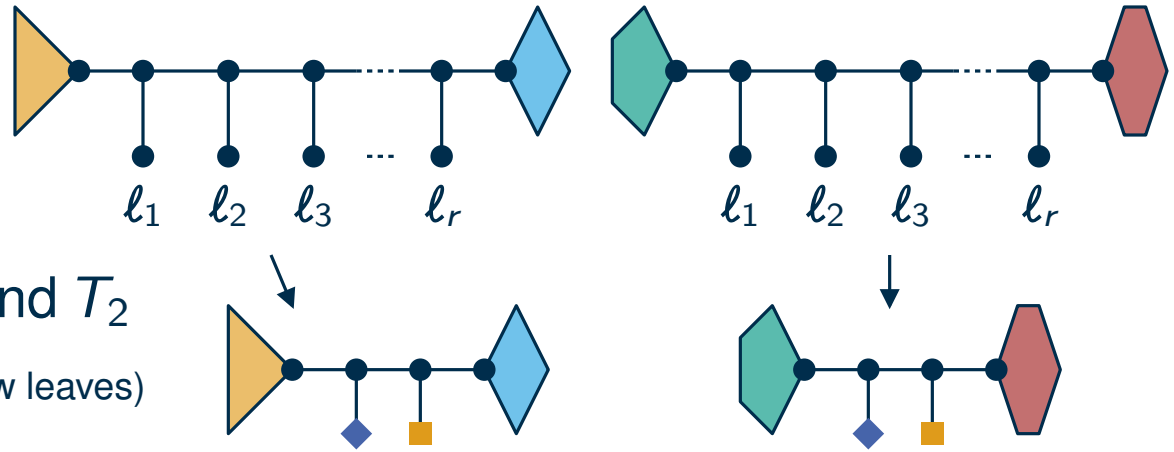
- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length two (two new leaves)

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length two (two new leaves)

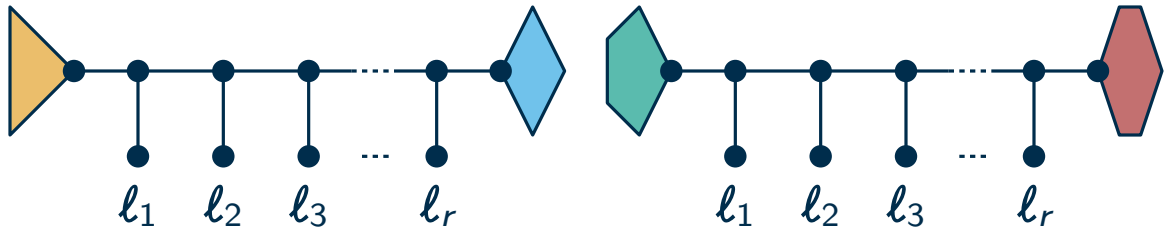
Is this rule safe?

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

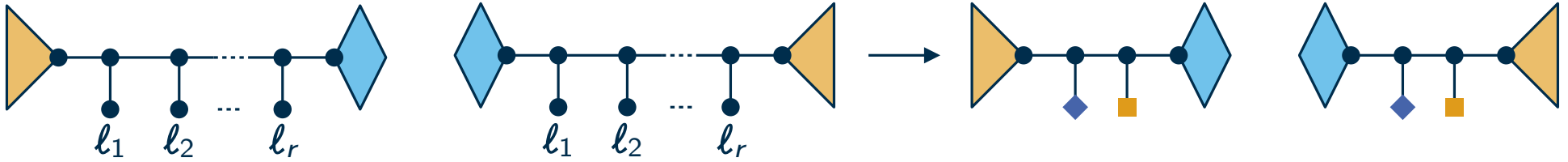
- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length two (two new leaves)

Problem

- number of required trees may be reduced

Is this rule safe?

requires three trees



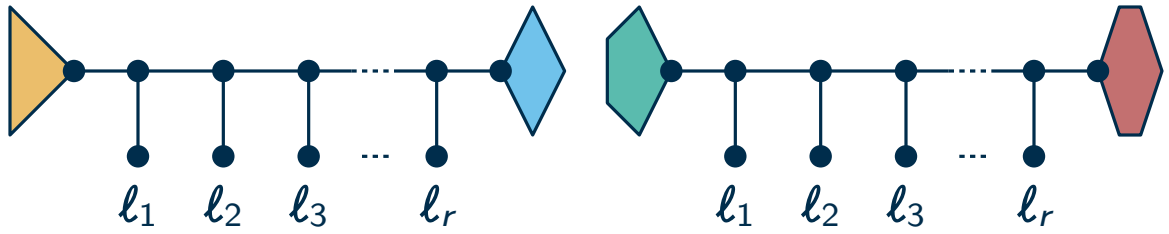
two trees suffice

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

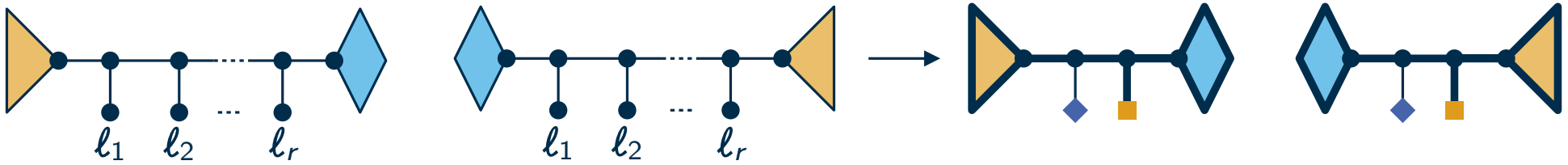
- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length two (two new leaves)

Problem

- number of required trees may be reduced

Is this rule safe?

requires three trees



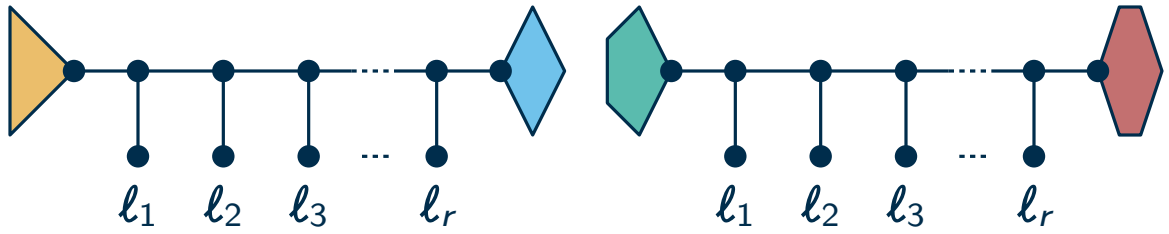
two trees suffice

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



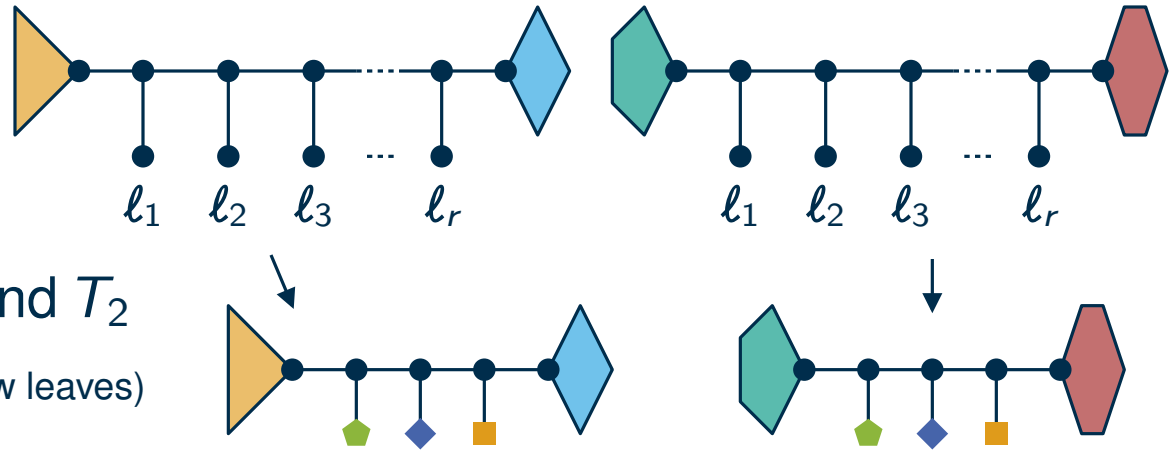
Reduction rule 2

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

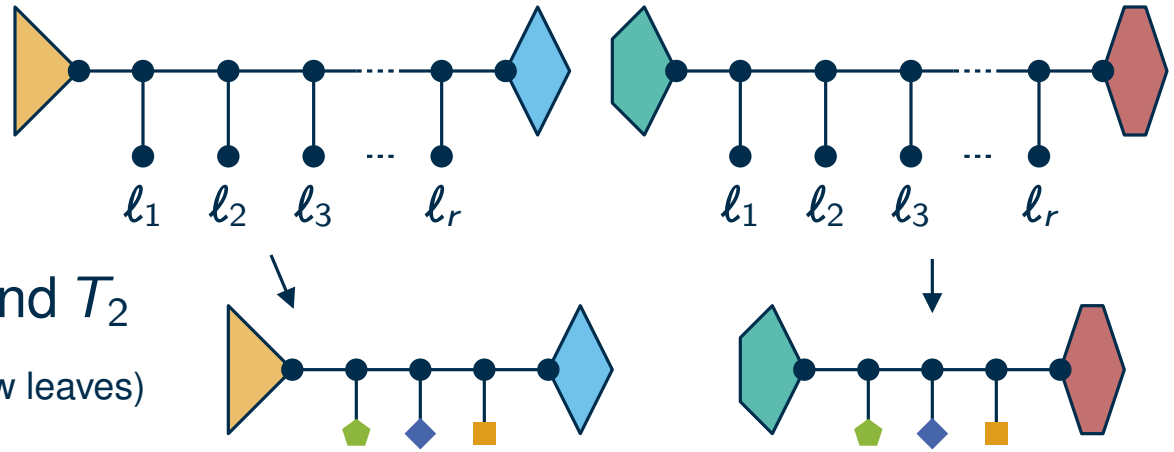
- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length three (three new leaves)

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length three (three new leaves)

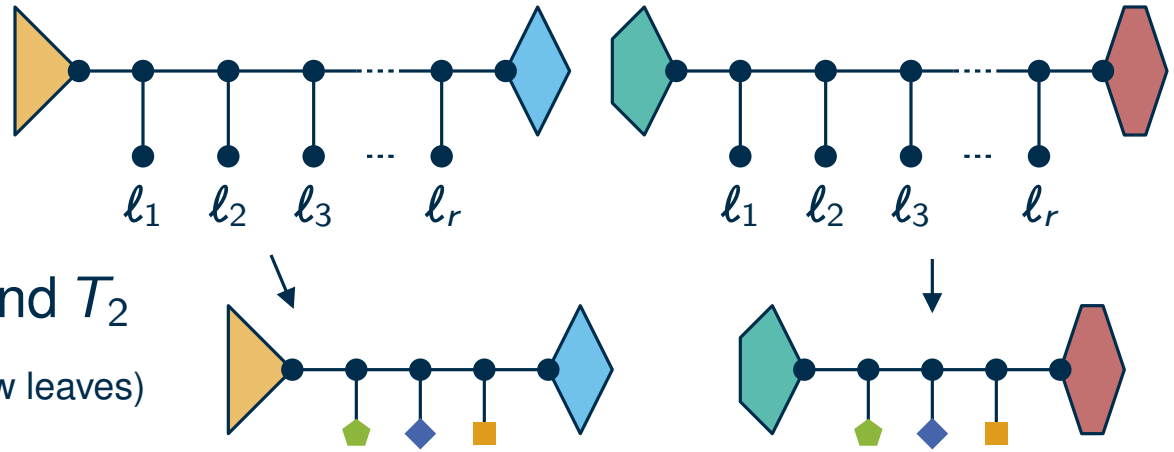
Lemma: Reduction rule 2 is safe and can be applied in polynomial time.

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length three (three new leaves)

Lemma: Reduction rule 2 is safe and can be applied in polynomial time.

Proof

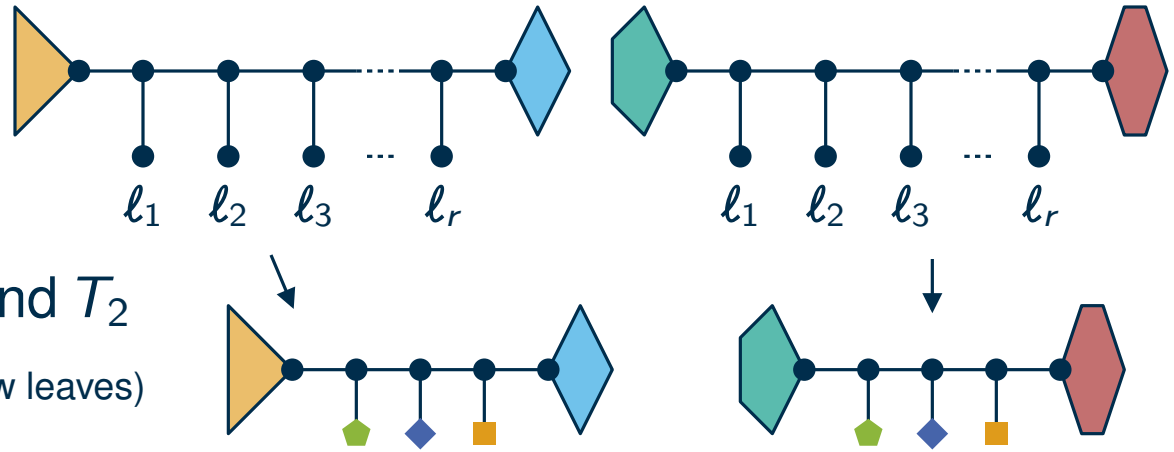
- case distinction (not here; but not very difficult)

Undesirable tree structures 2

Claim: Every tree of the agreement forest has only few leaves. (after applying reduction rule 1)

Counterexample

- many leaves in a shared subtree



Reduction rule 2

- find path with attached leaves $\ell_1, \ldots, \ell_r$ in T_1 and T_2
- replace it with a path of length three (three new leaves)

Lemma: Reduction rule 2 is safe and can be applied in polynomial time.

Proof

- case distinction (not here; but not very difficult)
- polynomial running time: for example by iteratively shortening paths of length 4

Undesirable tree structures 3

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

Undesirable tree structures 3

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Undesirable tree structures 3

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Is this really a problem?

Undesirable tree structures 3

amortized

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Amortized perspective

- after every three leaves, we see different subtrees

Is this really a problem?

Undesirable tree structures 3

amortized

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Is this really a problem?

Amortized perspective

- after every three leaves, we see different subtrees
- each of them yields at least one tree in the agreement forest

(if a, b, c, d, ... are all in the same tree)

Undesirable tree structures 3

amortized

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Is this really a problem?

Amortized perspective

- after every three leaves, we see different subtrees
- each of them yields at least one tree in the agreement forest (if a, b, c, d, . . . are all in the same tree)
- few trees in the agreement forest $\Rightarrow$ few leaves

Undesirable tree structures 3

amortized

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Is this really a problem?

Amortized perspective

- after every three leaves, we see different subtrees
- each of them yields at least one tree in the agreement forest (if $a, b, c, d, \ldots$ are all in the same tree)
- few trees in the agreement forest $\Rightarrow$ few leaves

Are we sure, there are no other undesirable structures?

Undesirable tree structures 3

amortized

Claim: Every tree of the agreement forest has only few leaves. (after applying rules 1 and 2)

Counterexample

- many leaves in a shared subtree

Is this really a problem?

Amortized perspective

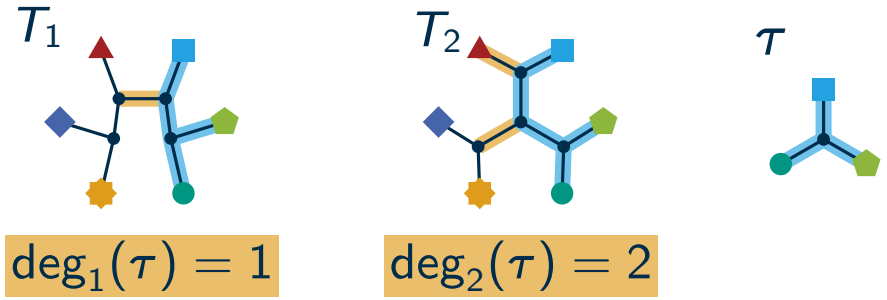
- after every three leaves, we see different subtrees
- each of them yields at least one tree in the agreement forest (if $a, b, c, d, \ldots$ are all in the same tree)
- few trees in the agreement forest $\Rightarrow$ few leaves

Lemma
Exhaustively applying reduction rules 1 and 2 to a yes-instance yields an equivalent instance with at most ck leaves (for a small constant c).

Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node

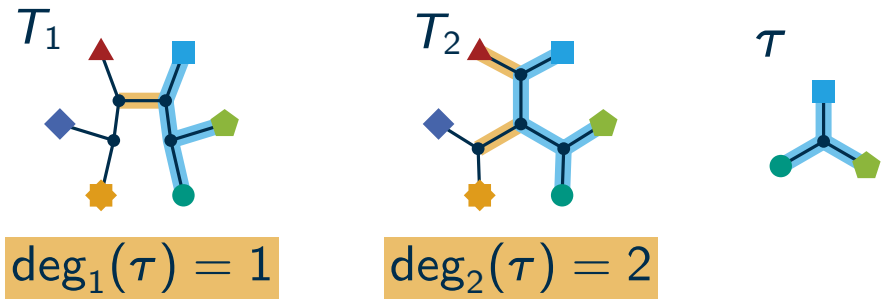


Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node

Amortized perspective (as before, but more formal)



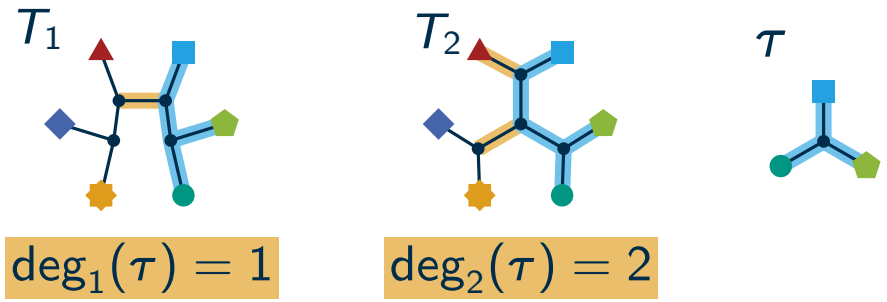
Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node

Amortized perspective (as before, but more formal)

- τ splits T_i into $\deg_i(\tau)$ subtrees



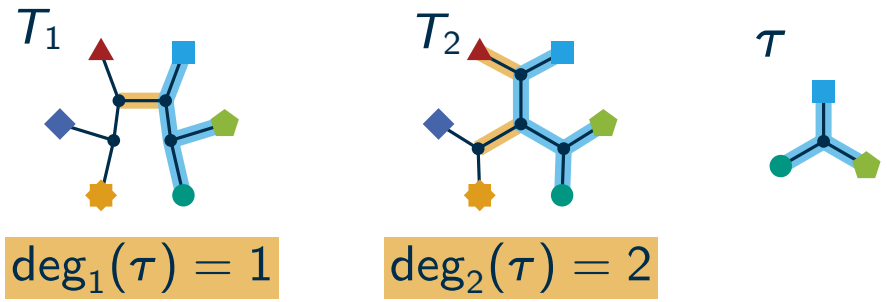
Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node

Amortized perspective (as before, but more formal)

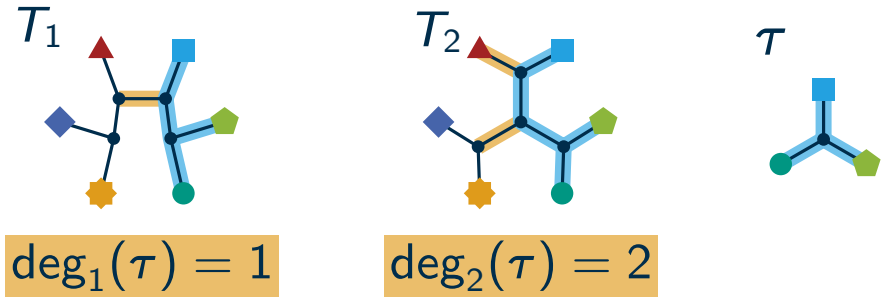
- τ splits T_i into $\deg_i(\tau)$ subtrees
- it follows: big degree $\deg_i(\tau) \Rightarrow$ many trees in F



Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node



Amortized perspective (as before, but more formal)

- τ splits T_i into $\deg_i(\tau)$ subtrees
- it follows: big degree $\deg_i(\tau) \Rightarrow$ many trees in F

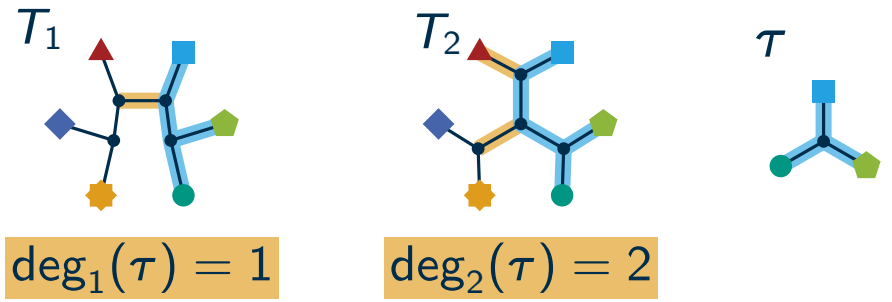
Lemma (few trees $\Rightarrow$ small degrees)

If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Rough plan of action

The degree of a subtree

- contract a tree τ of F in T_i to a single node ($i \in \{1, 2\}$)
- let $\deg_i(\tau)$ be the degree of the resulting node



Amortized perspective (as before, but more formal)

- τ splits T_i into $\deg_i(\tau)$ subtrees
- it follows: big degree $\deg_i(\tau) \Rightarrow$ many trees in F

Lemma (few trees $\Rightarrow$ small degrees)
If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

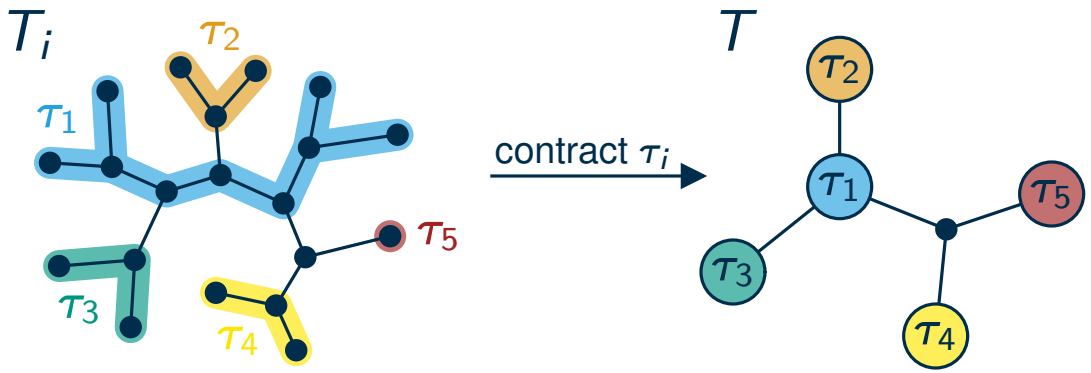
The sum of degrees is low

Lemma (few trees $\Rightarrow$ small degrees)

If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Proof

- contract each τ of F in $T_i \rightarrow$ tree T



The sum of degrees is low

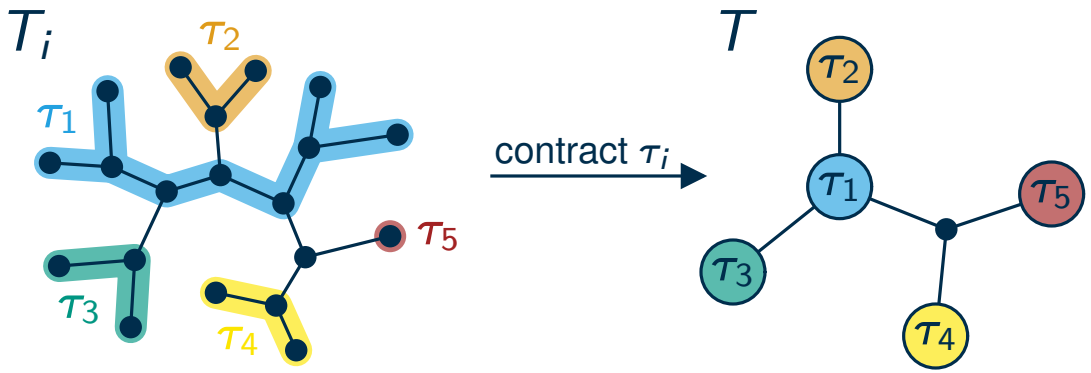
Lemma (few trees $\Rightarrow$ small degrees)

If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Proof

- contract each τ of F in $T_i \rightarrow$ tree T
- summing all degrees in T:

$$\sum_{v \in V(T)} \deg(v) = 2m = 2n - 2$$



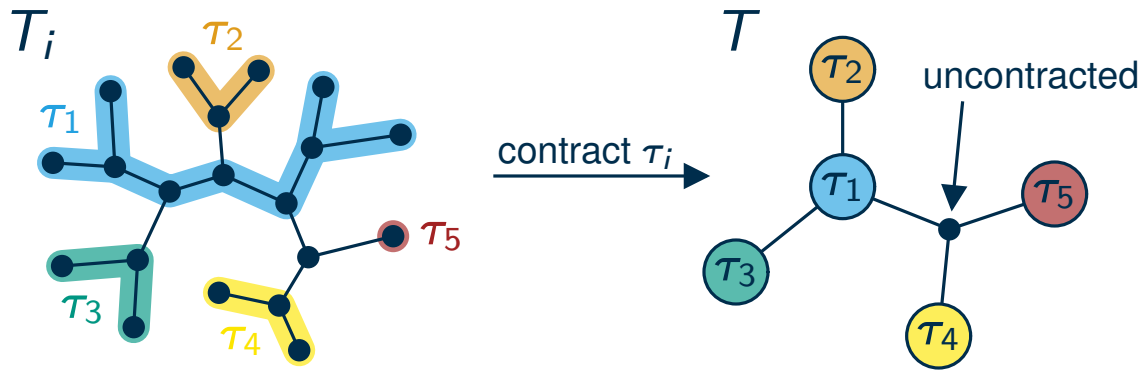
The sum of degrees is low

Lemma (few trees $\Rightarrow$ small degrees)
If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Proof

- contract each τ of F in $T_i \rightarrow$ tree T
- summing all degrees in T:

$$\sum_{v \in V(T)} \deg(v) = 2m = 2n - 2$$



- vertex v of T is either: one of the k trees τ_i or an uncontracted vertex of T_i with degree 3

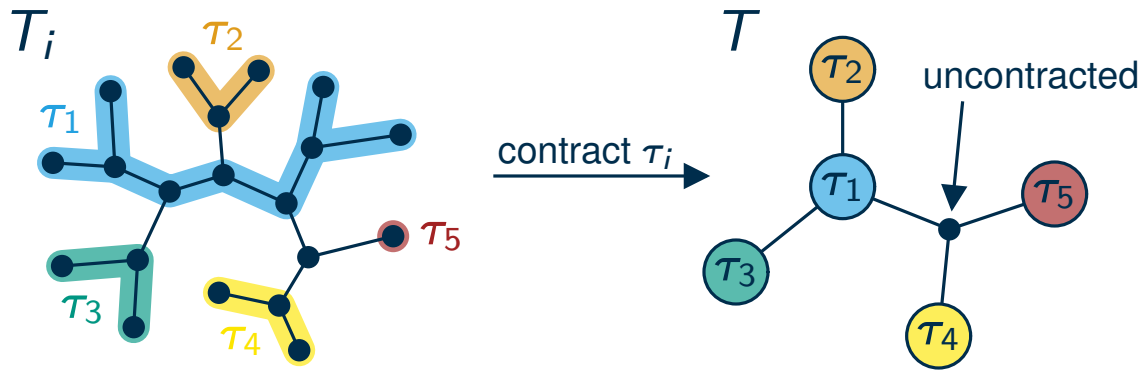
The sum of degrees is low

Lemma (few trees $\Rightarrow$ small degrees)
If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Proof

- contract each τ of F in $T_i \rightarrow$ tree T
- summing all degrees in T:

$$\sum_{v \in V(T)} \deg(v) = 2m = 2n - 2$$



- vertex v of T is either: one of the k trees τ_i or an uncontracted vertex of T_i with degree 3
- let n_3 be the number of uncontracted vertices; then:

$$3n_3 + \sum_{\tau \in F} \deg_i(\tau) = 2(k + n_3) - 2$$

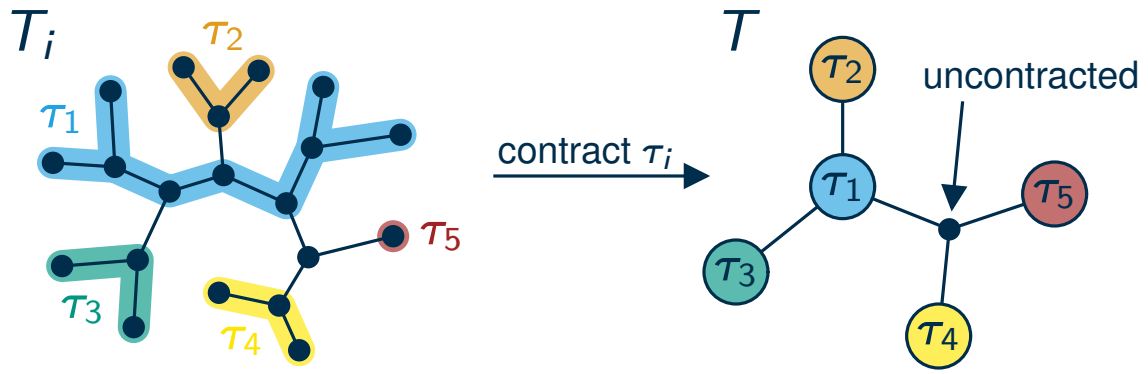
The sum of degrees is low

Lemma (few trees $\Rightarrow$ small degrees)
If F consists of k trees, then $\sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$.

Proof

- contract each τ of F in $T_i \rightarrow$ tree T
- summing all degrees in T:

$$\sum_{v \in V(T)} \deg(v) = 2m = 2n - 2$$

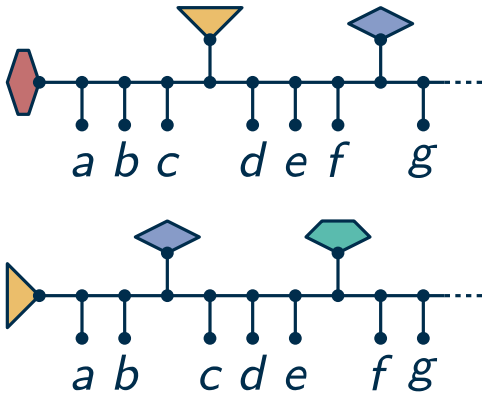


- vertex v of T is either: one of the k trees τ_i or an uncontracted vertex of T_i with degree 3
- let n_3 be the number of uncontracted vertices; then:

$$3n_3 + \sum_{\tau \in F} \deg_i(\tau) = 2(k + n_3) - 2 \quad \Rightarrow \quad n_3 + \sum_{\tau \in F} \deg_i(\tau) = 2k - 2 \quad \Rightarrow \quad \sum_{\tau \in F} \deg_i(\tau) \leq 2k - 2$$

Trees of low degree have few leaves

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

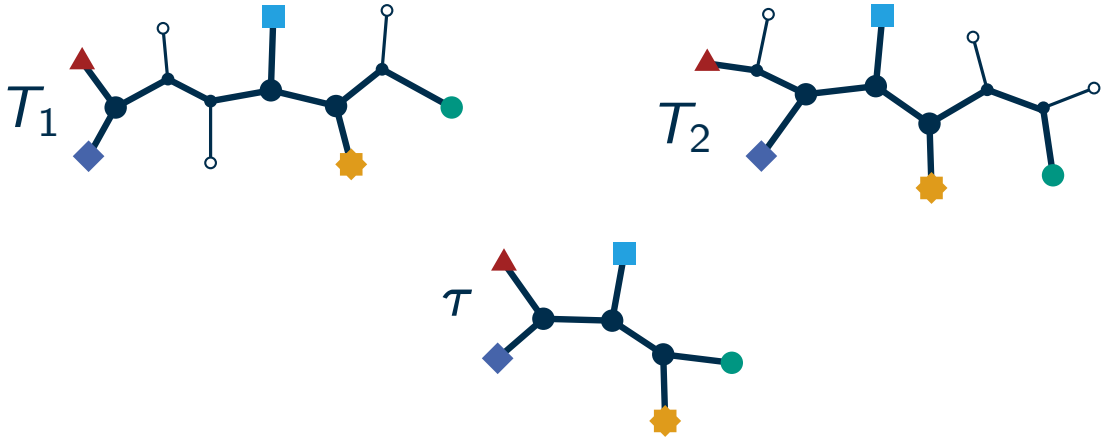


Trees of low degree have few leaves

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2

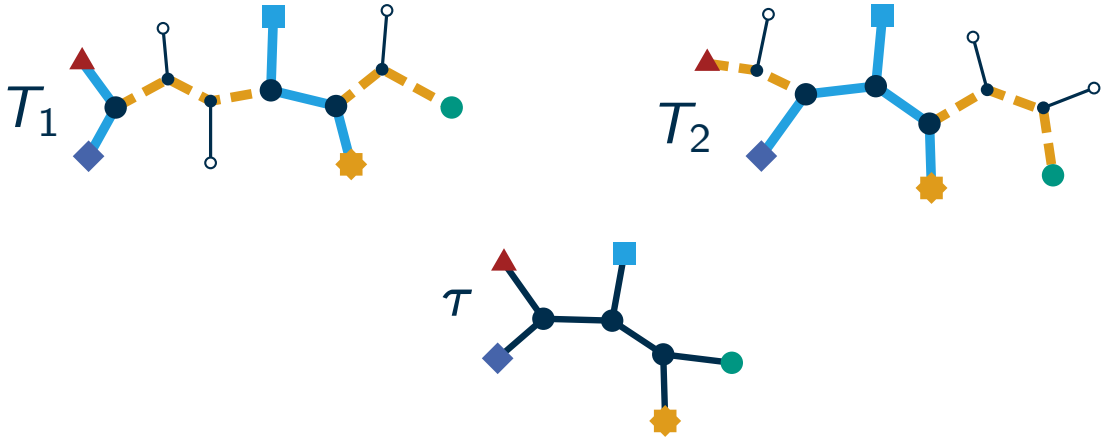


Trees of low degree have few leaves

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2
- each edge of τ is an edge or a path in T_i

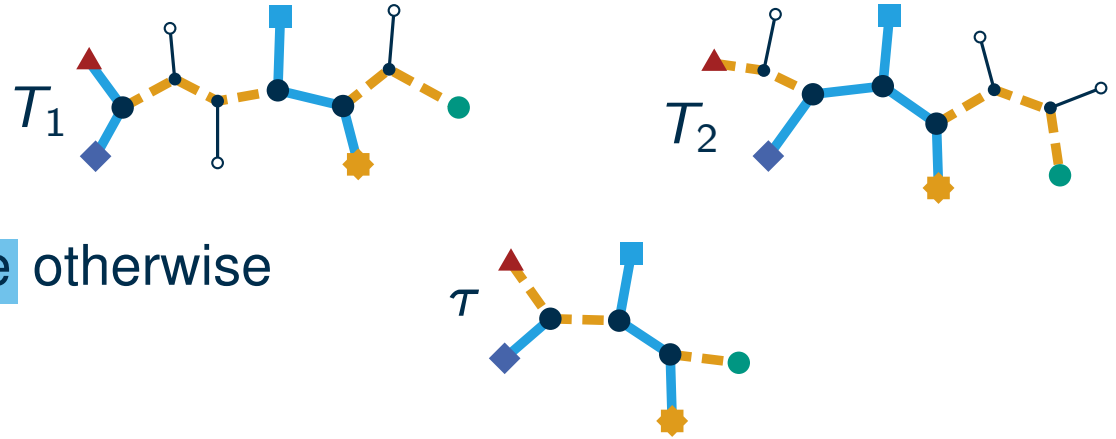


Trees of low degree have few leaves

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2
- each edge of τ is an edge or a path in T_i
- mark it as path if it is a path in T_1 or T_2 and edge otherwise

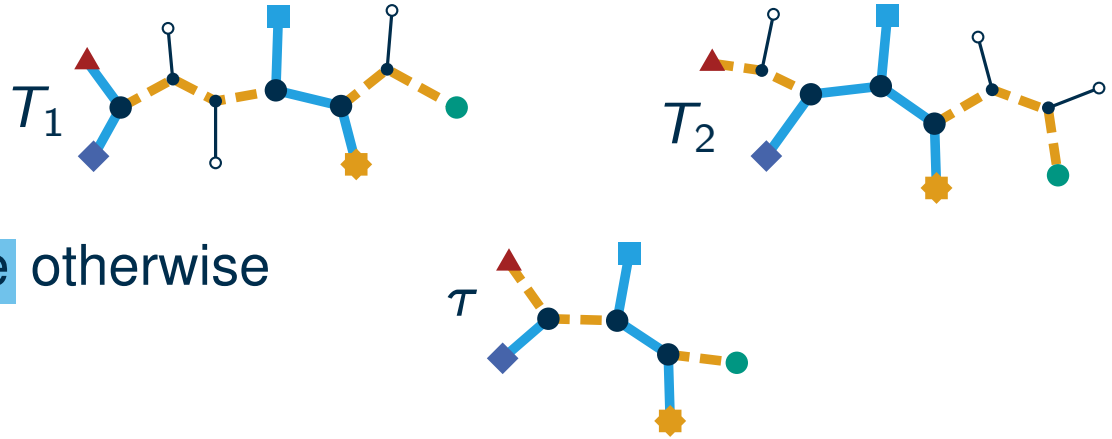


Trees of low degree have few leaves

Lemma (small degrees ⇒ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2
- each edge of τ is an edge or a path in T_i
- mark it as path if it is a path in T_1 or T_2 and edge otherwise



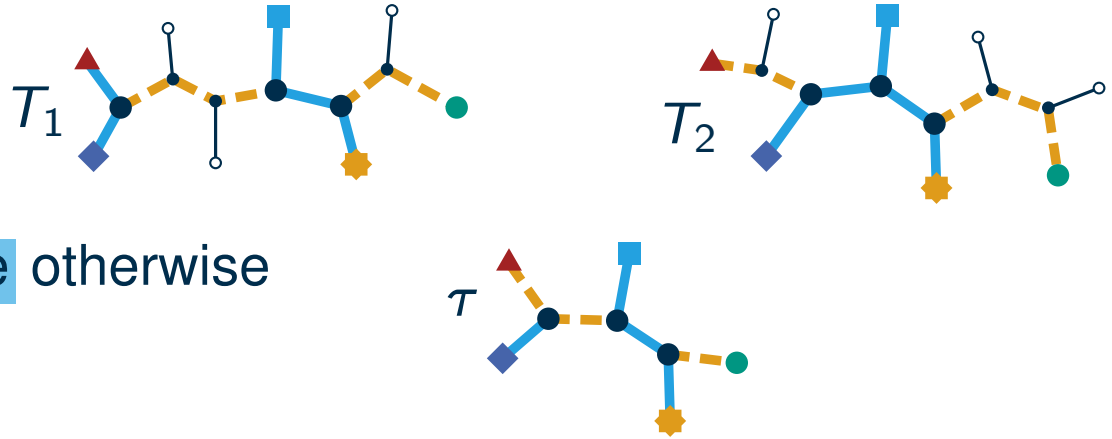
Why is the labelling useful?

Trees of low degree have few leaves

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2
- each edge of τ is an edge or a path in T_i
- mark it as path if it is a path in T_1 or T_2 and edge otherwise



Why is the labelling useful?

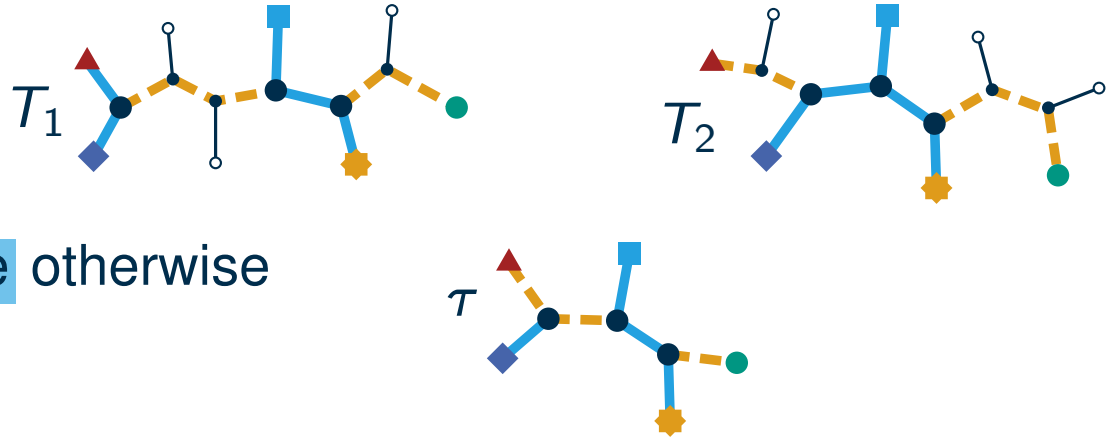
- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges

Trees of low degree have few leaves

Lemma (small degrees ⇒ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Two types of edges in τ

- consider how τ lies in T_1 and T_2
- each edge of τ is an edge or a path in T_i
- mark it as path if it is a path in T_1 or T_2 and edge otherwise

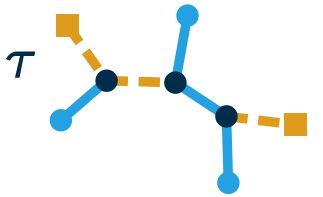


Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges

Two types of leaves in τ

- classify each leaf by its incident edge: path leaf or edge leaf

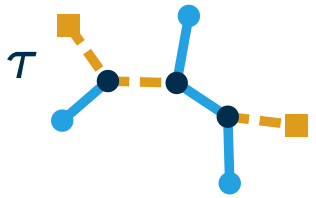


How do the reduction rules help?

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges

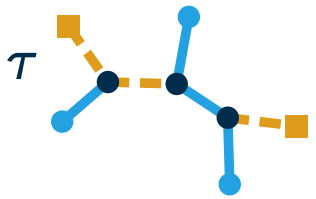


How do the reduction rules help?

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges



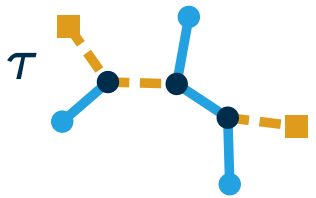
What do the reduction rules tell us about τ?

How do the reduction rules help?

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges



What do the reduction rules tell us about τ?

- rule 1: no cherry with two edge leaves

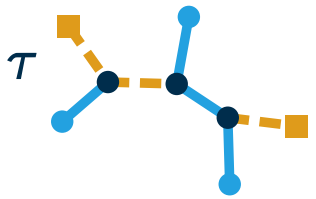
would be reduced by rule 1

How do the reduction rules help?

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges



What do the reduction rules tell us about τ?

- rule 1: no cherry with two edge leaves
- rule 2: no “leave-path” of length 4 with only edge edges

would be reduced by rule 1

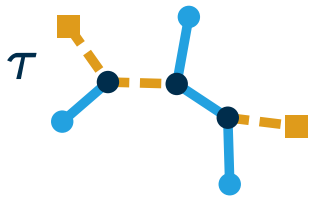
would be reduced by rule 2

How do the reduction rules help?

Lemma (small degrees $\Rightarrow$ few leaves)
Every tree $\tau \in F$ has at most $c(\deg_1(\tau) + \deg_2(\tau))$ leaves (after applying rules 1 and 2).

Why is the labelling useful?

- #path edges $\leq \deg_1(\tau) + \deg_2(\tau)$
- it remains to show: #leaves $\leq c \cdot$ #path edges



What do the reduction rules tell us about τ?

- rule 1: no cherry with two edge leaves
- rule 2: no "leave-path" of length 4 with only edge edges

would be reduced by rule 1

would be reduced by rule 2

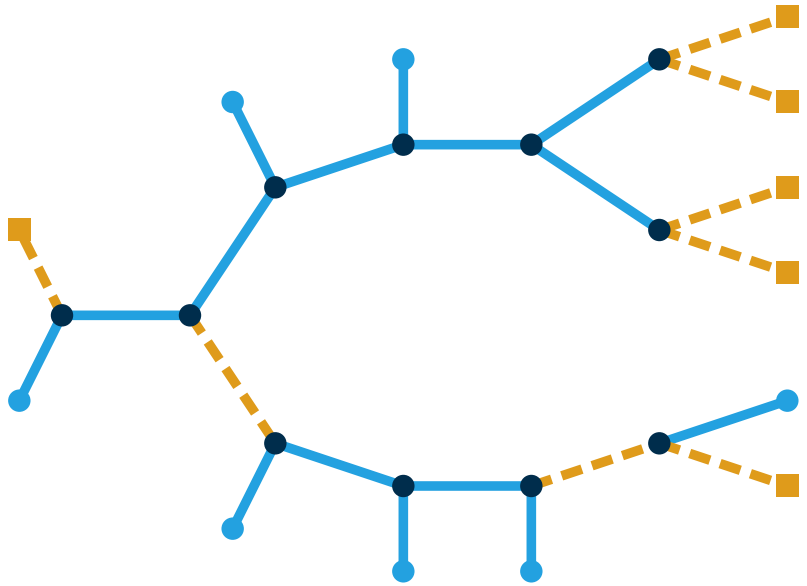
Lemma (implies the above lemma)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Here comes the proof

forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof



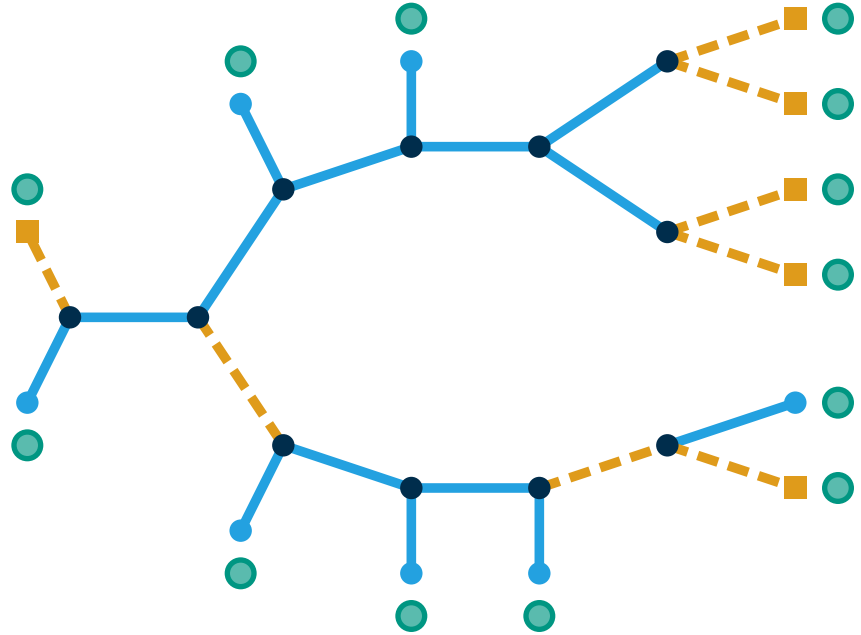
Here comes the proof

forbidden structures

Lemma (implies: small degrees $\Rightarrow$ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token per leaf $\rightarrow$ count



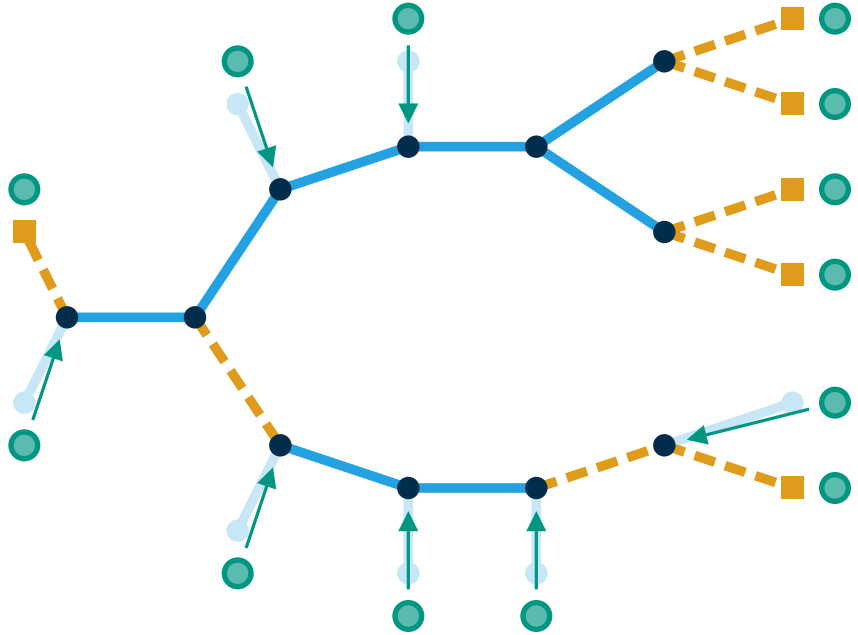
Here comes the proof

forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token per leaf → count
- delete blue leaves and move to neighbor



Here comes the proof

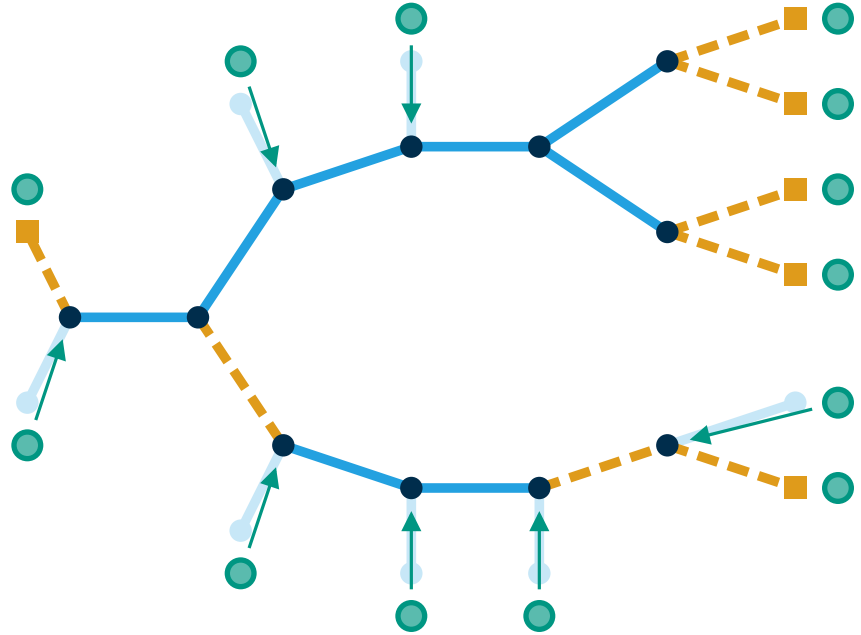
forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex

Why?



Here comes the proof

forbidden structures

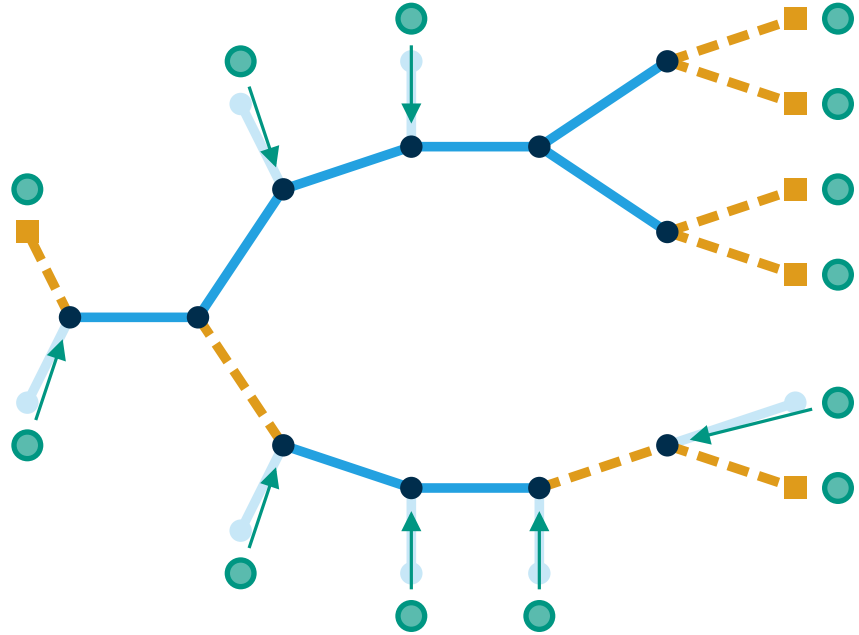
Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex

this is ok: each orange leaf is attached to an orange edge

we need to deal with this



Here comes the proof

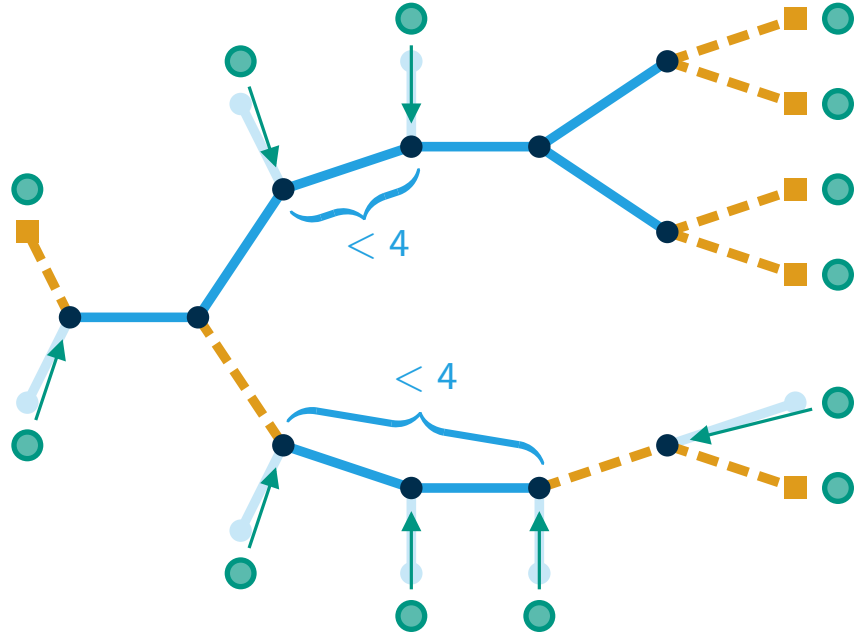
forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices

Why?



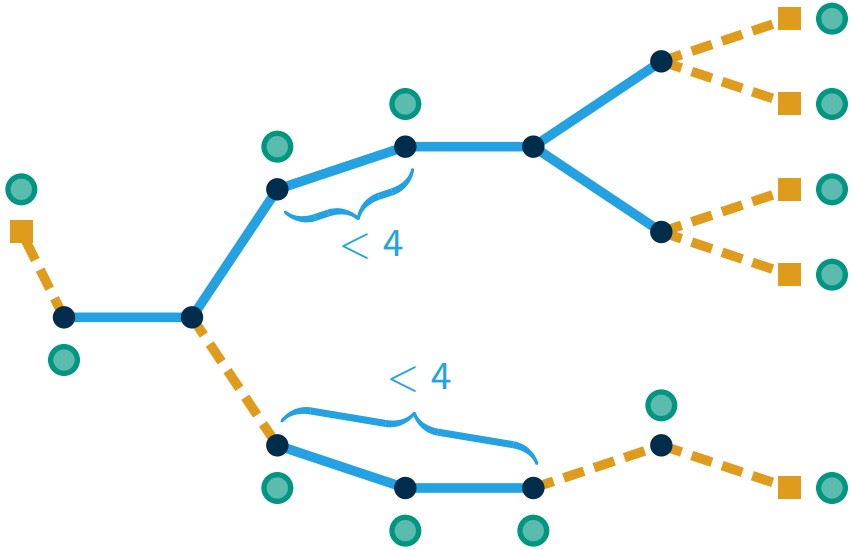
Here comes the proof

forbidden structures

Lemma (implies: small degrees $\Rightarrow$ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token per leaf $\rightarrow$ count
- delete blue leaves and move to neighbor
  - only orange leaves and 1 per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices



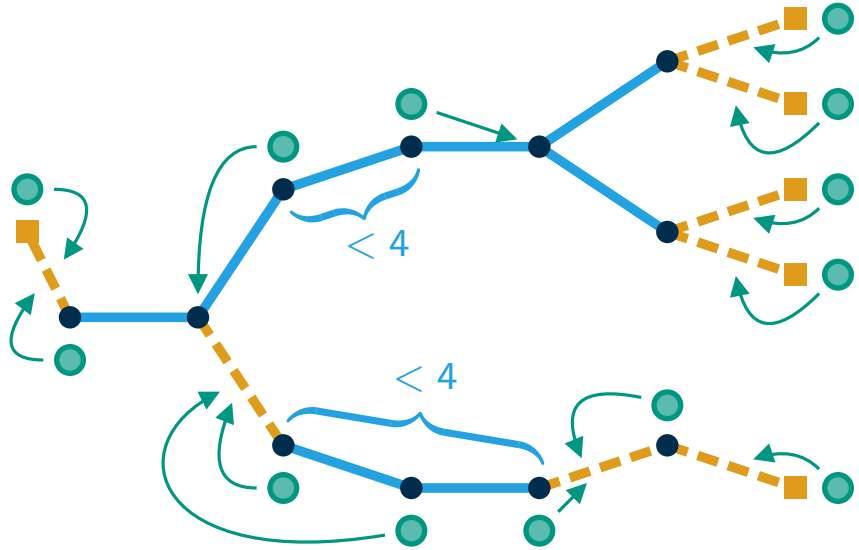
Here comes the proof

forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]



Here comes the proof

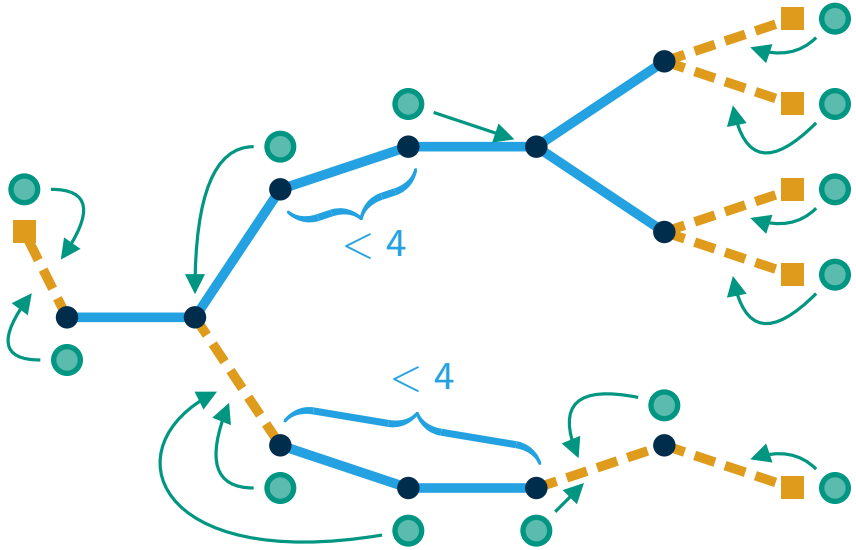
forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]
  - ≤ 6 ● per deg-3 vertex and ≤ 4 ● per orange edge

Why?



Here comes the proof

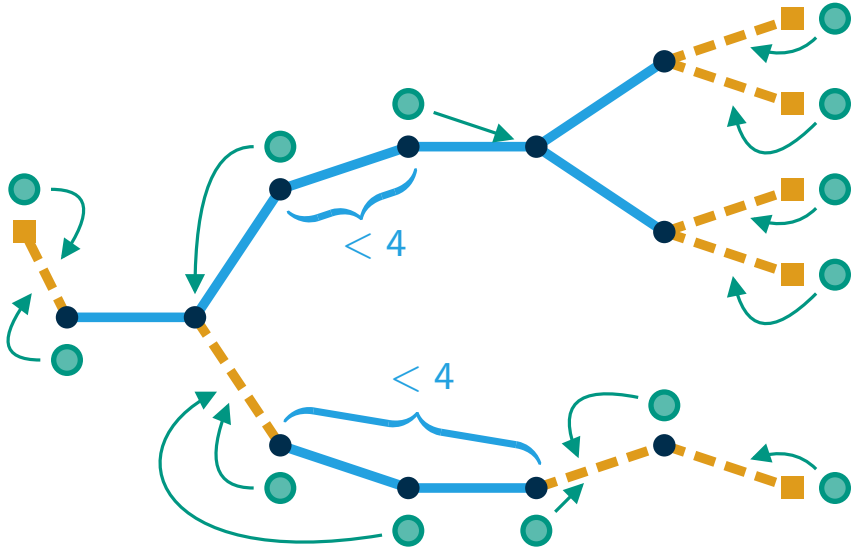
forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]
  - ≤ 6 ● per deg-3 vertex and ≤ 4 ● per orange edge
- #(deg-3 vertices) $<$ #(leaves)

(exercise sheet: $n_3 = n_1 - 2$)



Here comes the proof

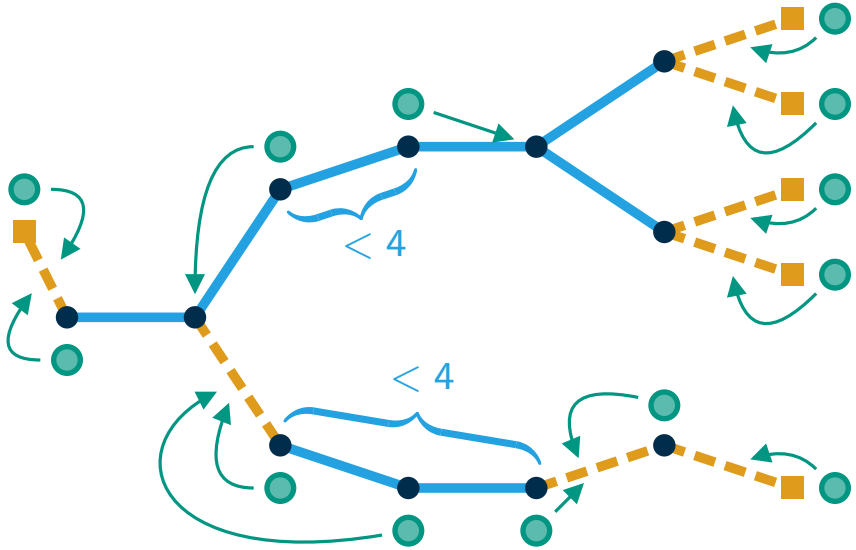
forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]
  - ≤ 6 ● per deg-3 vertex and ≤ 4 ● per orange edge
- #(deg-3 vertices) $<$ #(leaves) $=$ #orange leaves

(all leaves are orange)



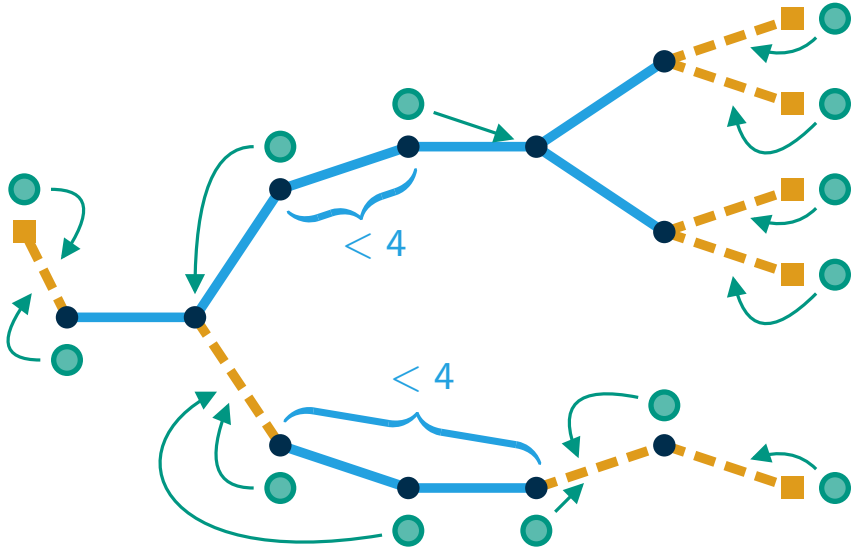
Here comes the proof

forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]
  - ≤ 6 ● per deg-3 vertex and ≤ 4 ● per orange edge
- #(deg-3 vertices) $<$ #(leaves) $=$ #orange leaves $\leq m_r$



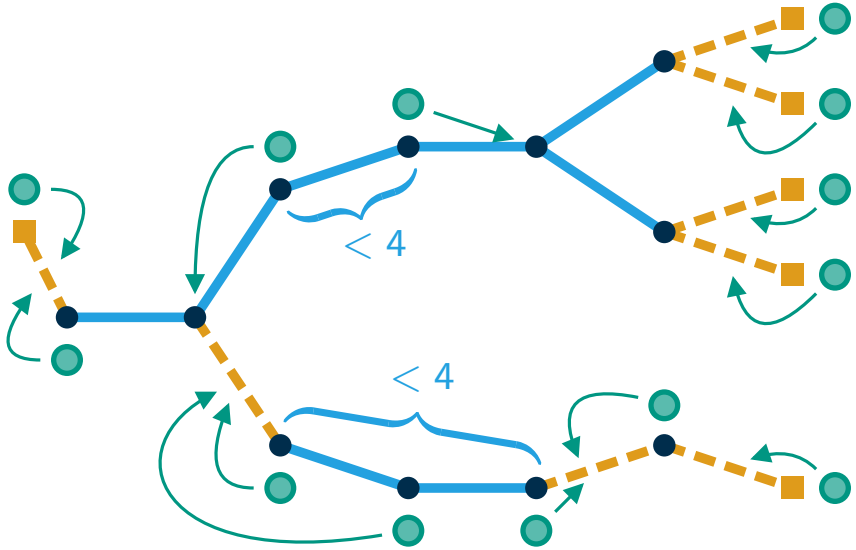
Here comes the proof

forbidden structures

Lemma (implies: small degrees ⇒ few leaves)
Let τ be a full binary tree with m_r orange edges (non-orange edges are blue). If τ does not contain one of the two forbidden structures, then it has at most $10m_r$ leaves.

Proof

- start with one token ● per leaf → count ●
- delete blue leaves and move ● to neighbor
  - only orange leaves and 1● per leaf and deg-2 vertex
  - blue paths of degree-2 vertices have < 4 vertices
- move each ● to closest [deg-3 vertex or orange edge]
  - ≤ 6 ● per deg-3 vertex and ≤ 4 ● per orange edge
- #(deg-3 vertices) $<$ #(leaves) $=$ #orange leaves $\leq m_r$
- #(leaves of τ) $=$ #● $\leq 6m_r + 4m_r$

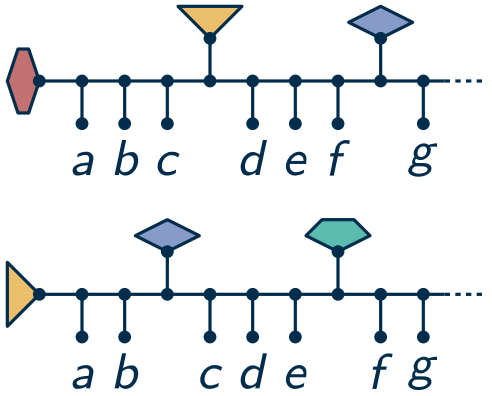


What have we done?

Lemma
Exhaustively applying reduction rules 1 and 2 to a yes-instance yields an equivalent instance with at most ck leaves (for a small constant c).

Proof

- just seen: number of leaves of $\tau \in F$ is at most $10(\deg_1(\tau) + \deg_2(\tau))$
  - τ has many leaves $\Rightarrow$ it splits T_1 or T_2 into many subtrees

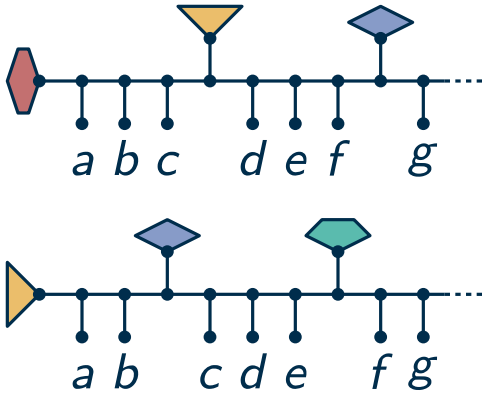


What have we done?

Lemma
Exhaustively applying reduction rules 1 and 2 to a yes-instance yields an equivalent instance with at most ck leaves (for a small constant c).

Proof

- just seen: number of leaves of $\tau \in F$ is at most $10(\deg_1(\tau) + \deg_2(\tau))$
  - τ has many leaves $\Rightarrow$ it splits T_1 or T_2 into many subtrees
- seen before: $\sum_{\tau \in F} \deg_i(\tau) < 2k$
  - $\tau \in F$ split T_1 or T_2 into many subtrees $\Rightarrow$ F has many trees

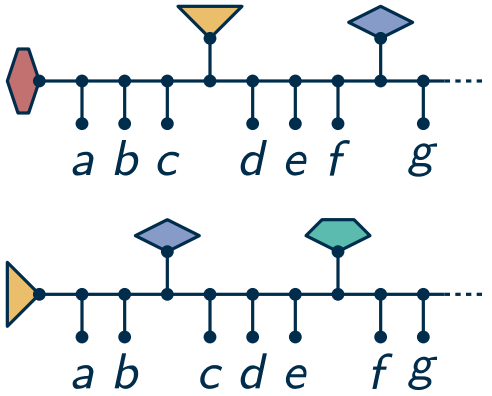


What have we done?

Lemma
Exhaustively applying reduction rules 1 and 2 to a yes-instance yields an equivalent instance with at most ck leaves (for a small constant c).

Proof

- just seen: number of leaves of $\tau \in F$ is at most $10(\deg_1(\tau) + \deg_2(\tau))$
  - τ has many leaves $\Rightarrow$ it splits T_1 or T_2 into many subtrees
- seen before: $\sum_{\tau \in F} \deg_i(\tau) < 2k$
  - $\tau \in F$ split T_1 or T_2 into many subtrees $\Rightarrow$ F has many trees
- it follows: F has at most $40k$ leaves
  - many leaves $\Rightarrow$ many trees

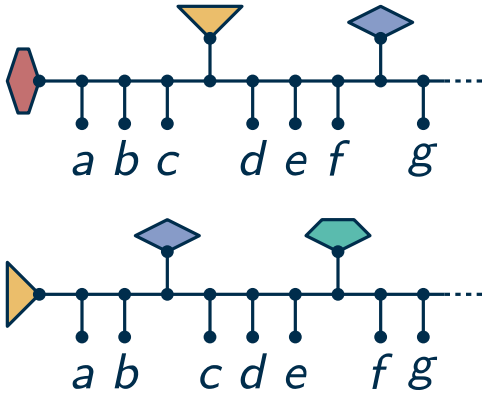


What have we done?

Lemma
Exhaustively applying reduction rules 1 and 2 to a yes-instance yields an equivalent instance with at most ck leaves (for a small constant c).

Proof

- just seen: number of leaves of $\tau \in F$ is at most $10(\deg_1(\tau) + \deg_2(\tau))$
  - τ has many leaves $\Rightarrow$ it splits T_1 or T_2 into many subtrees
- seen before: $\sum_{\tau \in F} \deg_i(\tau) < 2k$
  - $\tau \in F$ split T_1 or T_2 into many subtrees $\Rightarrow$ F has many trees
- it follows: F has at most $40k$ leaves
  - many leaves $\Rightarrow$ many trees



Reduction rule 3: more than $40k$ leaves after applying rule 1 and 2 $\rightarrow$ no-instance of size $O(1)$

Wrap-Up

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Theorem
MAXIMUM AGREEMENT FOREST has a kernel of size $O(k)$ that can be computed in poly time.

Wrap-Up

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Theorem
MAXIMUM AGREEMENT FOREST has a kernel of size $O(k)$ that can be computed in poly time.

Technique

- a specific goal beyond "I want a small kernel" helps (e.g., I want every tree in F to have few leaves)

Wrap-Up

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Theorem
MAXIMUM AGREEMENT FOREST has a kernel of size $O(k)$ that can be computed in poly time.

Technique

- a specific goal beyond "I want a small kernel" helps (e.g., I want every tree in F to have few leaves)
- counterexamples tell us how to reduce

Wrap-Up

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Theorem
MAXIMUM AGREEMENT FOREST has a kernel of size $O(k)$ that can be computed in poly time.

Technique

- a specific goal beyond "I want a small kernel" helps (e.g., I want every tree in F to have few leaves)
- counterexamples tell us how to reduce
- weakening the goal later might be helpful (e.g., amortizing over all trees in F)

Wrap-Up

Problem: MAXIMUM AGREEMENT FOREST
Given T_1, T_2, and a parameter k. Is there an agreement forest of at most k trees?

Theorem
MAXIMUM AGREEMENT FOREST has a kernel of size $O(k)$ that can be computed in poly time.

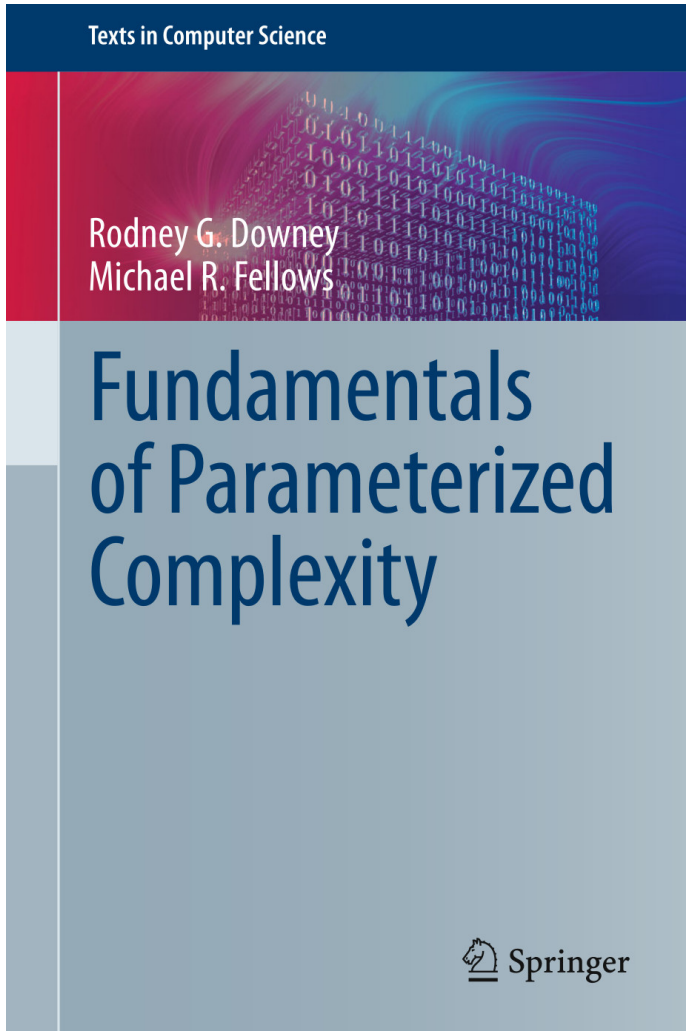
Technique

- a specific goal beyond "I want a small kernel" helps (e.g., I want every tree in F to have few leaves)
- counterexamples tell us how to reduce
- weakening the goal later might be helpful (e.g., amortizing over all trees in F)

Not seen today

- specific running time for kernelization
- specific running time for brute force on the kernel $O(4^k \cdot k^5)$ is possible

Literature



Comments

- chapter 4.10 is about the topic from today
- also contains links to the original publication
- freely available for KIT members

`link.springer.com/book/10.1007/978-1-4471-5559-1`