

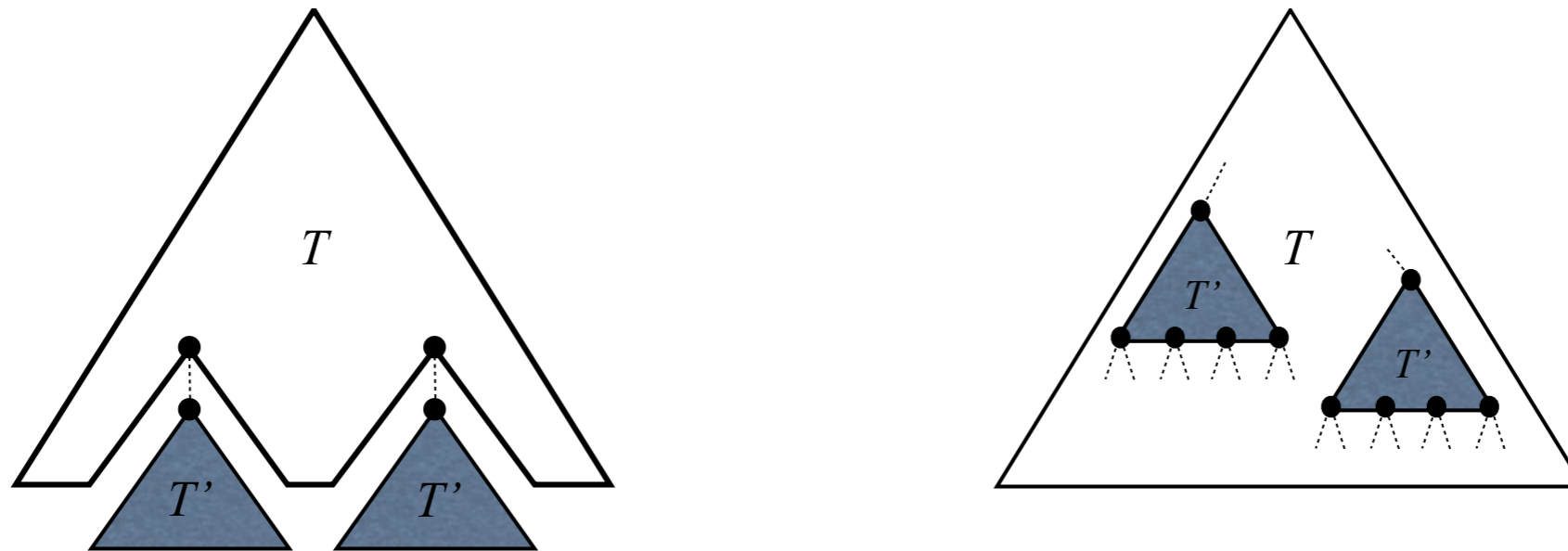
Tree Compression with Top Trees

Philip Bille, Inge Li Gørtz, Gad M. Landau, and Oren Weimann

Outline

- Tree Compression with repetitions
- Previous work
 - DAG compression
 - Tree grammar compression
- Top tree compression
 - Top trees and top tree compression
 - Compression analysis
 - Compressed navigation

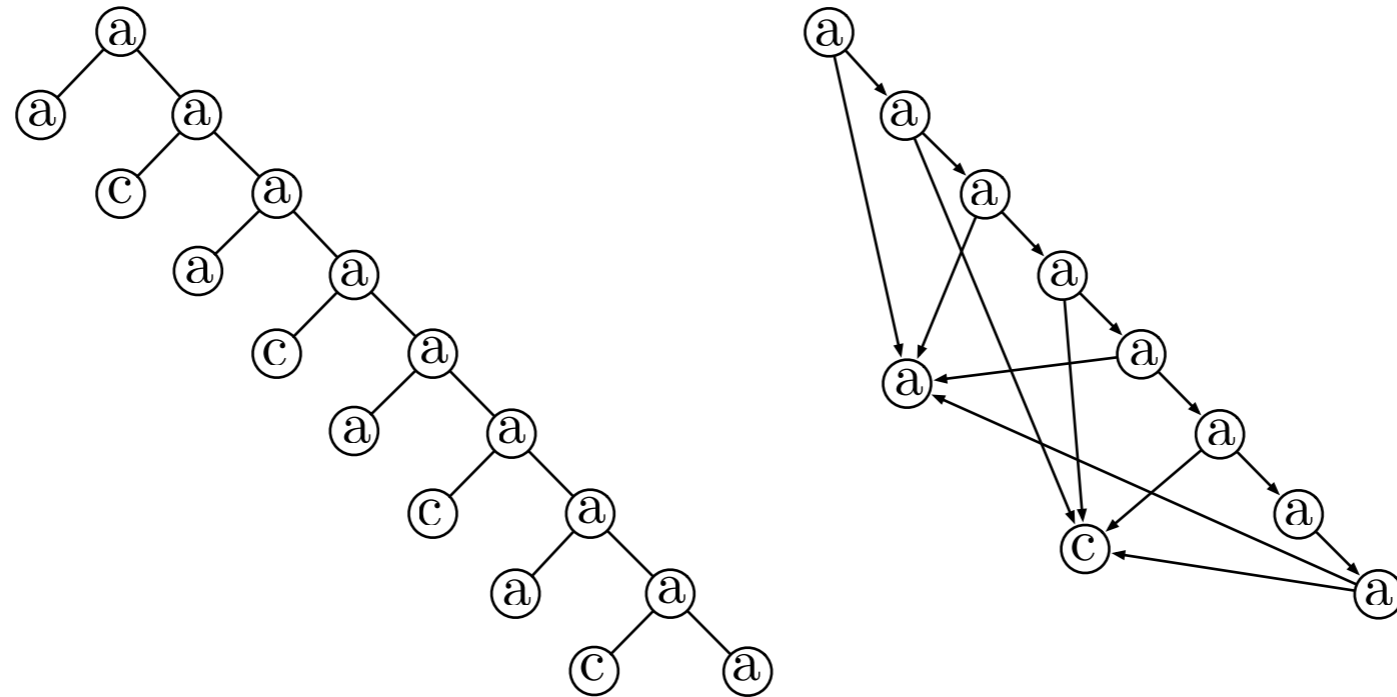
Tree Compression with Repetitions



- Let T be a labeled, rooted tree with N nodes over an alphabet of size σ .
- How to compress T in order to:
 - Take advantage of repetitions (*subtree repeats* or *tree pattern repeats*)
 - Obtain provably good guarantees on compression ratio.
 - Support efficient navigation (access, parent, depth, height, size, NCA, ...)

DAG Compression

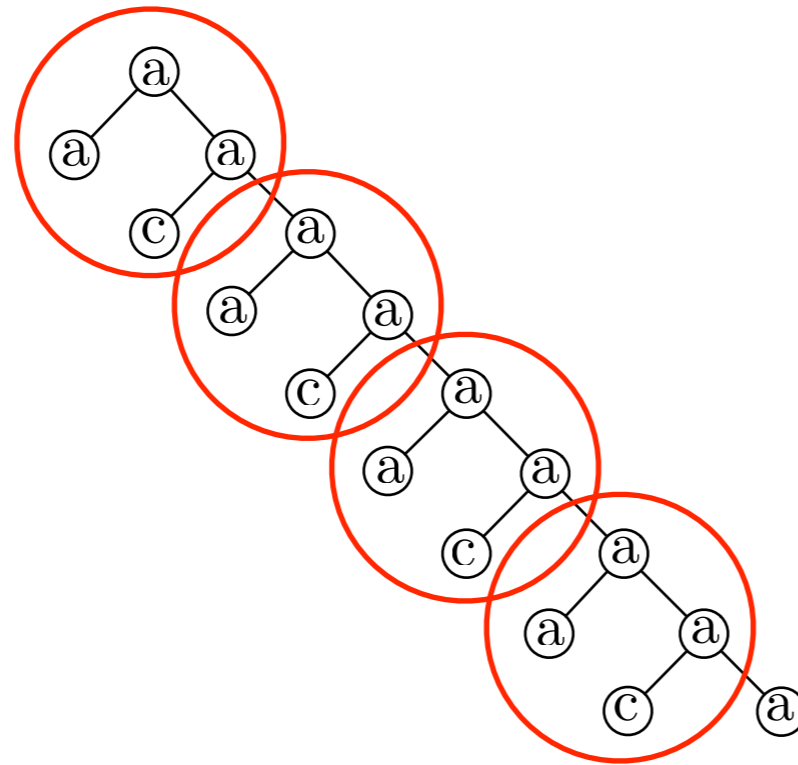
- Merge *subtree repeats* into directed acyclic graph (DAG) representing T.



- Takes advantage of subtree repeats but not tree pattern repeats.
- Smallest DAG is unique.
- We can build smallest DAG in $O(N)$ time [Downey, Sethi, Tarjan 1980]
- Smallest DAG can be exponentially smaller than N , but may not compress at all.
- We can support navigational operations in $O(\log N)$ time [B., Landau, Raman, Sadakane, Satti, Weimann 2011]
- Popular for XML compression. See e.g. [Buneman, Grohe, Koch 2003] [Frick, Grohe, Koch 2003]

Tree Grammars

- Encode *tree pattern repeats* using a grammar that generates T.



$$S \rightarrow A(A(A(A(a))))$$

$$A(x) \rightarrow a(a,a(c,x))$$

- Takes advantage of tree pattern repeats.
- NP-hard to find the tree smallest grammar (since even NP-hard for strings).
- Tree grammar can be exponentially smaller than the smallest DAG.
- We can support navigation in time proportional to height of grammar.
- Popular for XML compression. See e.g. [Busatto, Lohrey, Maneth 2004] [Maneth, Busatto 2004] [Lohrey, Maneth 2006] [Busatto, Lohrey, Maneth 2008] [Lohrey, Maneth, Mennicke 2010]

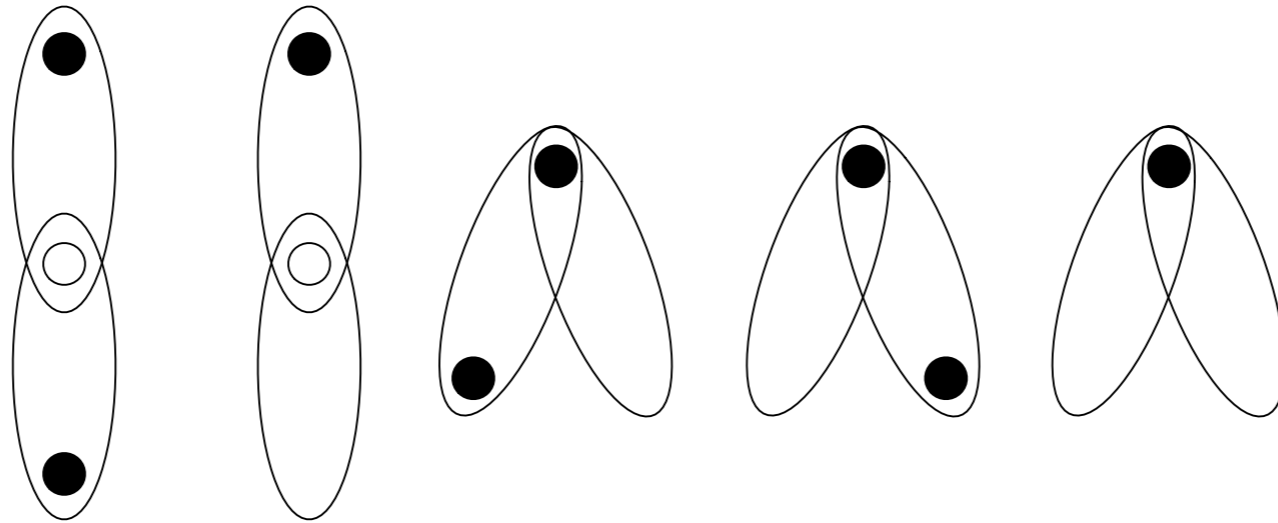
Top Tree Compression

- New and simple tree compression scheme:
 - Linear time construction.
 - Takes advantage of tree pattern repeats.
 - Compression ratio is always at least a factor of $(\log_{\sigma} N)^{0.19}$. Information theoretic worst-case lower bound is $\log_{\sigma} N$.
 - Can compress exponentially better than the smallest DAG.
 - Is never more than a $\log N$ factor larger than the smallest DAG
 - Navigational operations in $O(\log N)$ time.

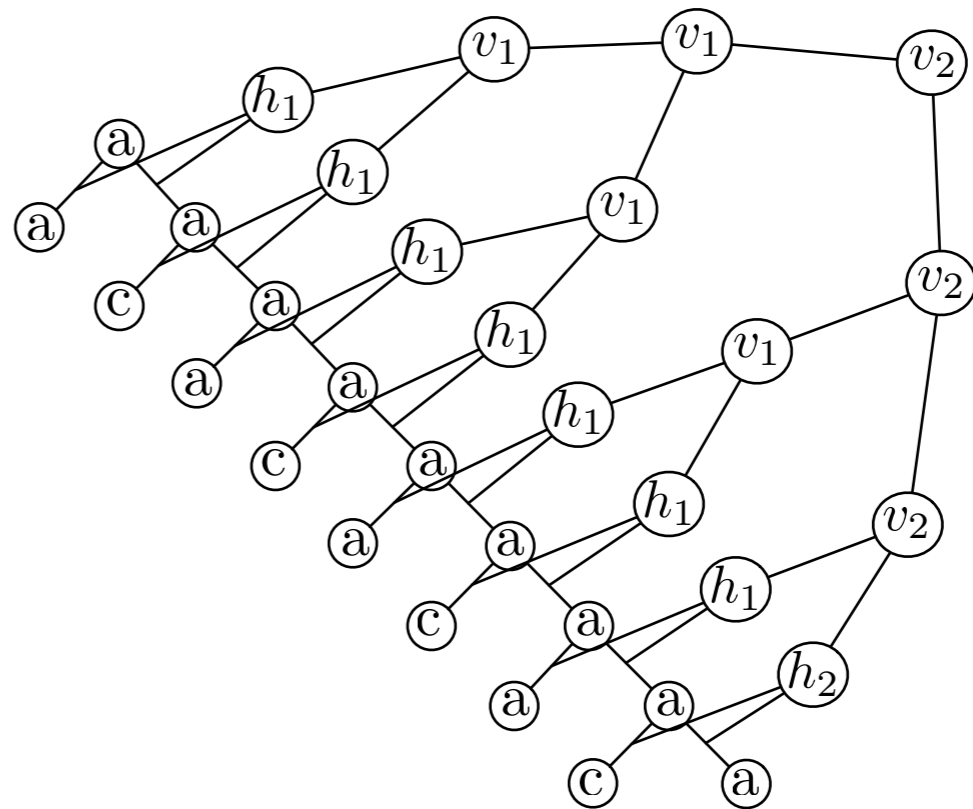
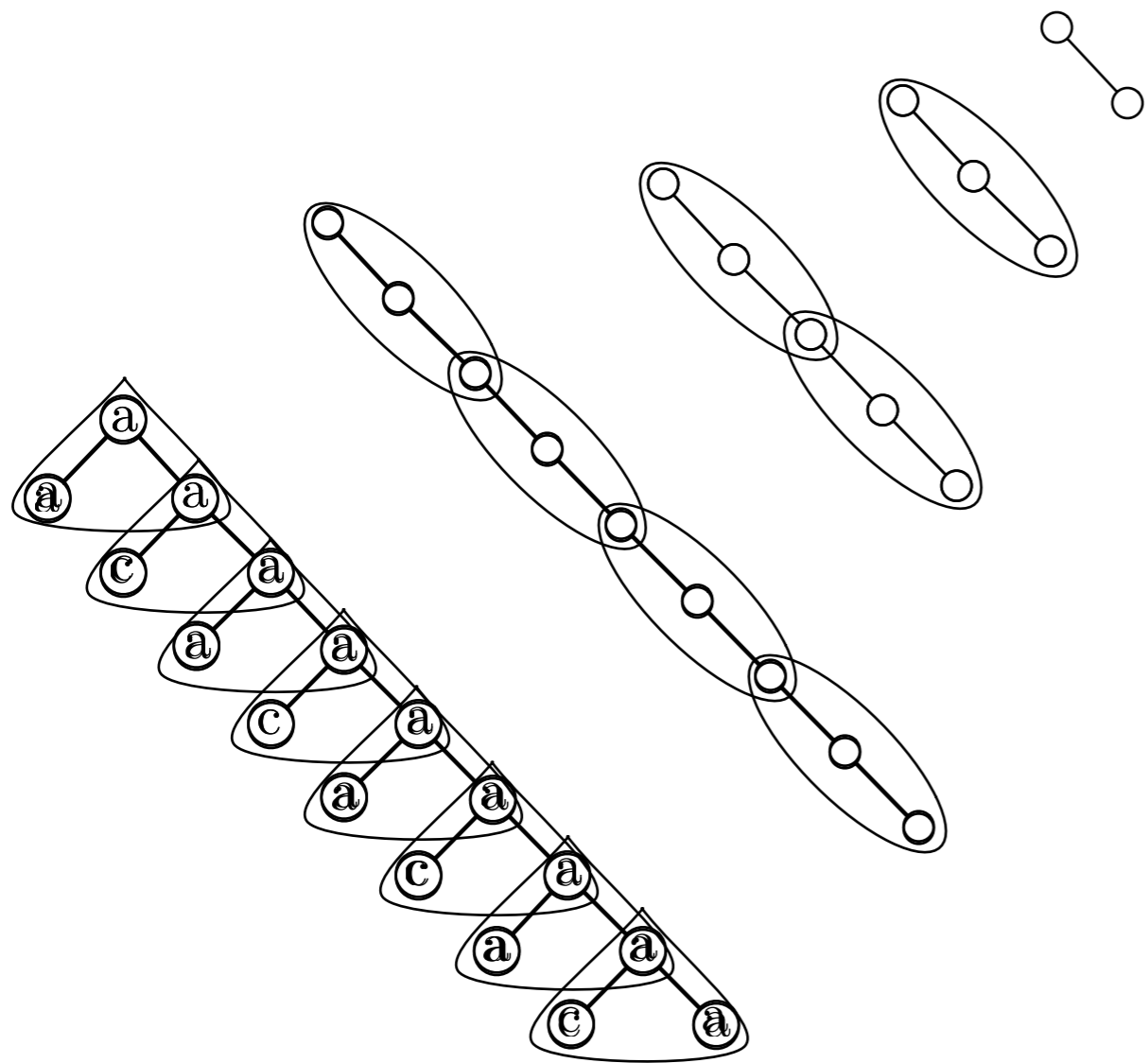
Top Trees

- Top tree for T is a decomposition of T into hierarchy of connected subtrees of T called *clusters*.
- Each cluster overlaps with adjacent clusters in 1 or 2 *boundary* nodes.

Top Tree Construction



- Start with edges of T as the bottom of cluster hierarchy (leaves of the top tree)
- Merge pairs of clusters greedily to form new clusters.
- Contract each clusters into an edge.
- Repeat until left with a single edge.

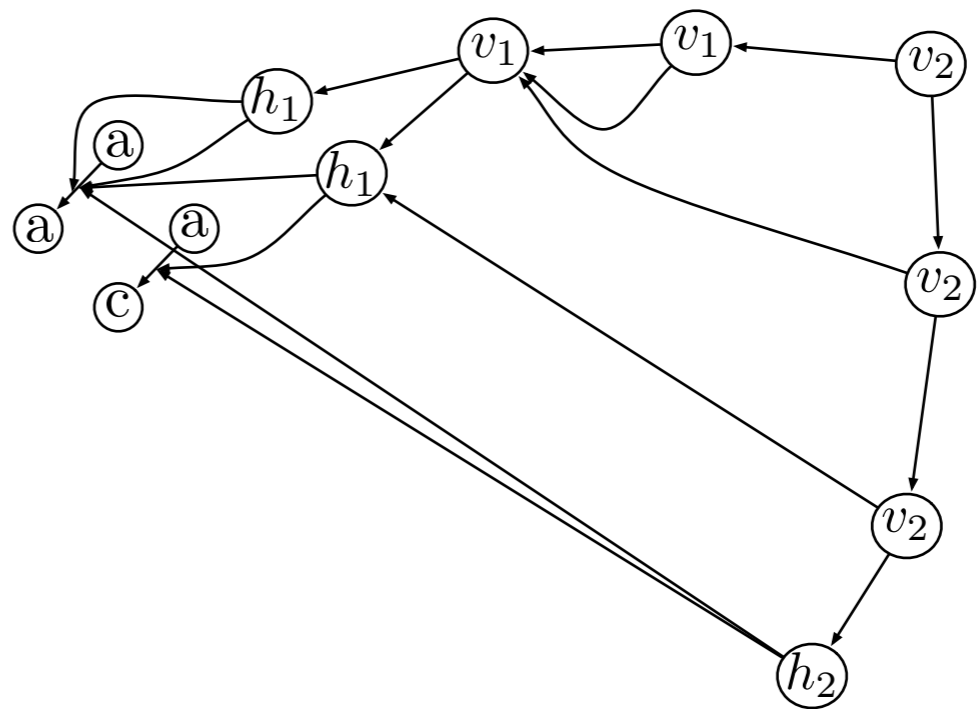
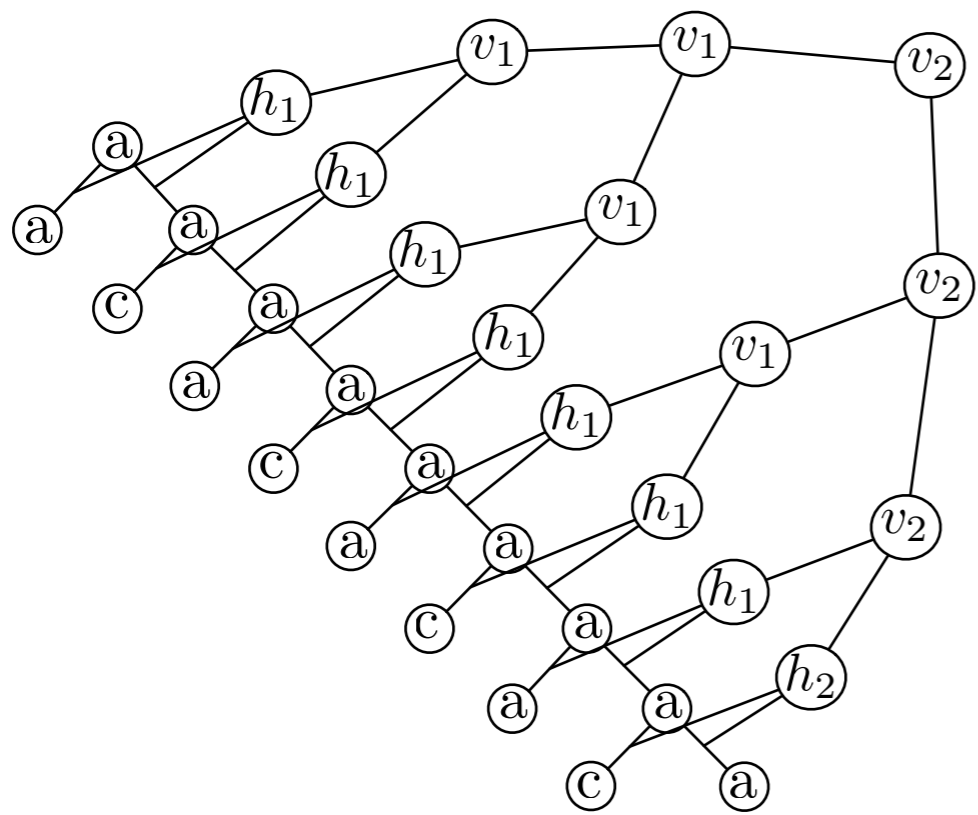


Top Tree Properties

- Top tree is a binary tree.
- Clusters size increases at each level by a factor of at most 2.
- Greedy merging decrease number of clusters a factor at least $8/7$.
- Size of top tree is $O(N)$
- Height of top tree is $O(\log N)$

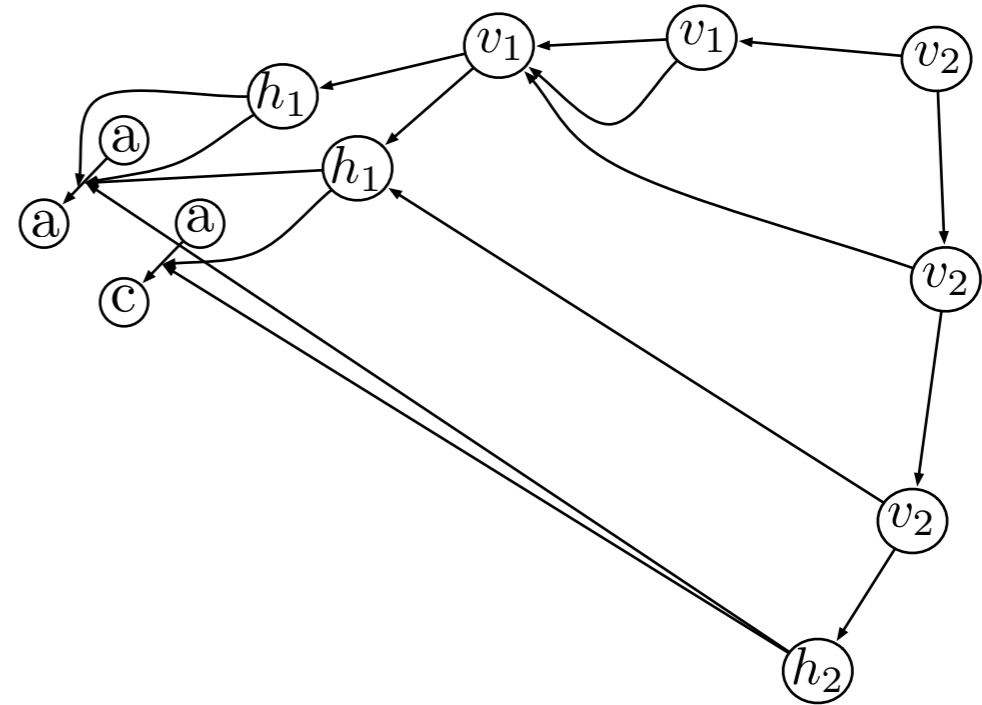
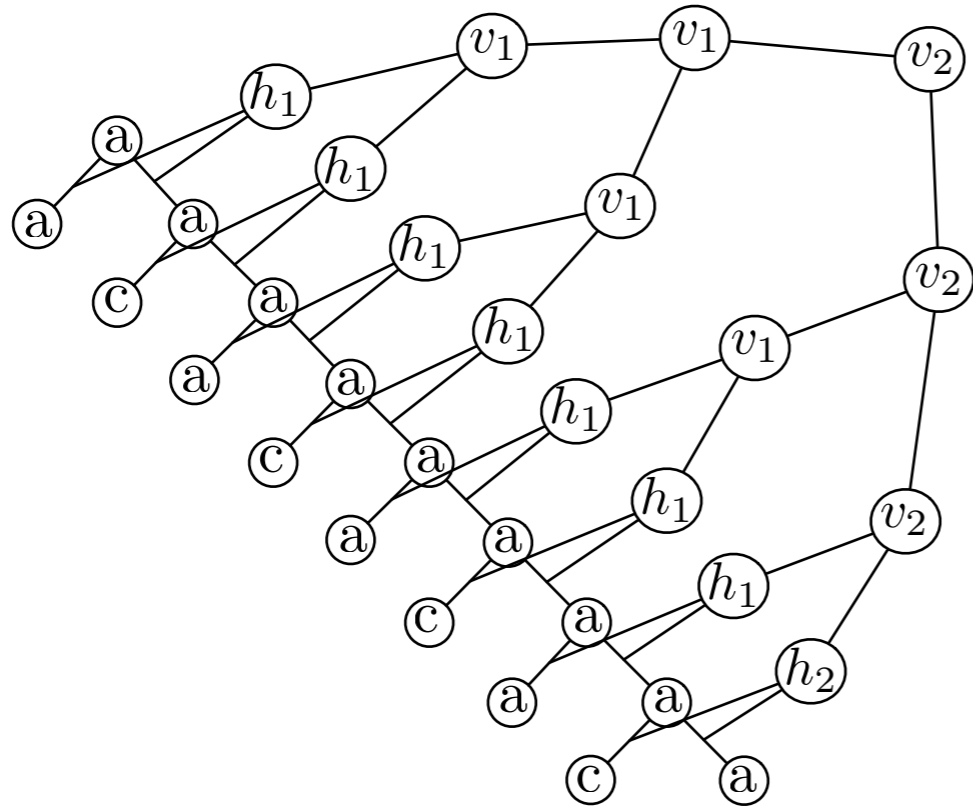
Top Tree Compression

- DAG compress top tree (!)
- Top tree compression may be viewed as *transformation* of input tree into another tree (which compresses well and a supports fast navigation).



Top Tree Compression Analysis

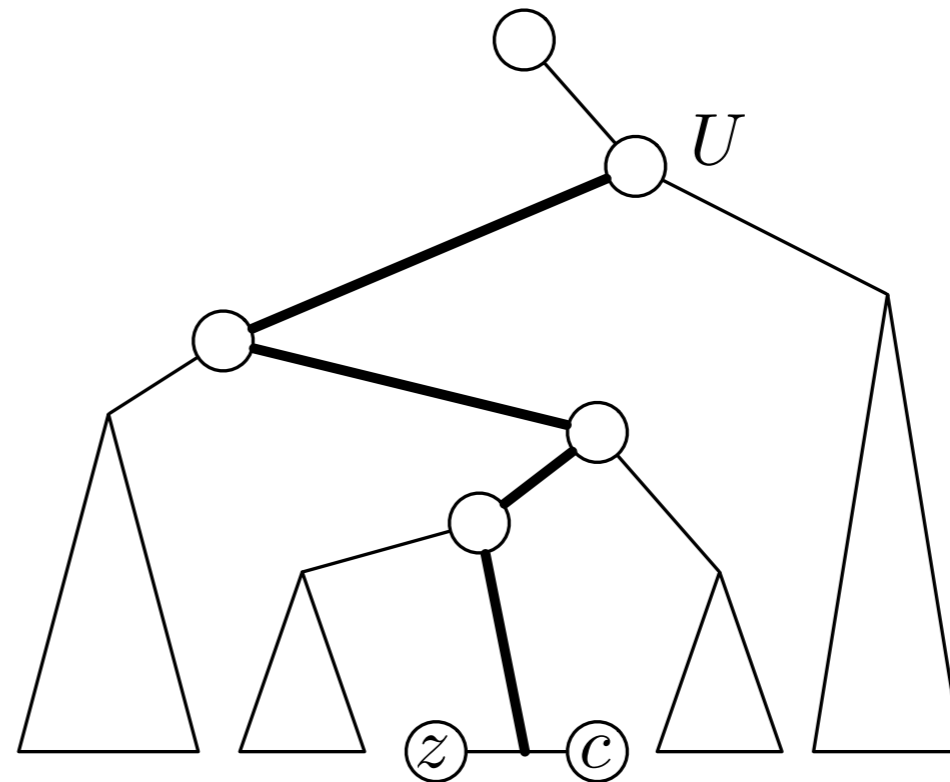
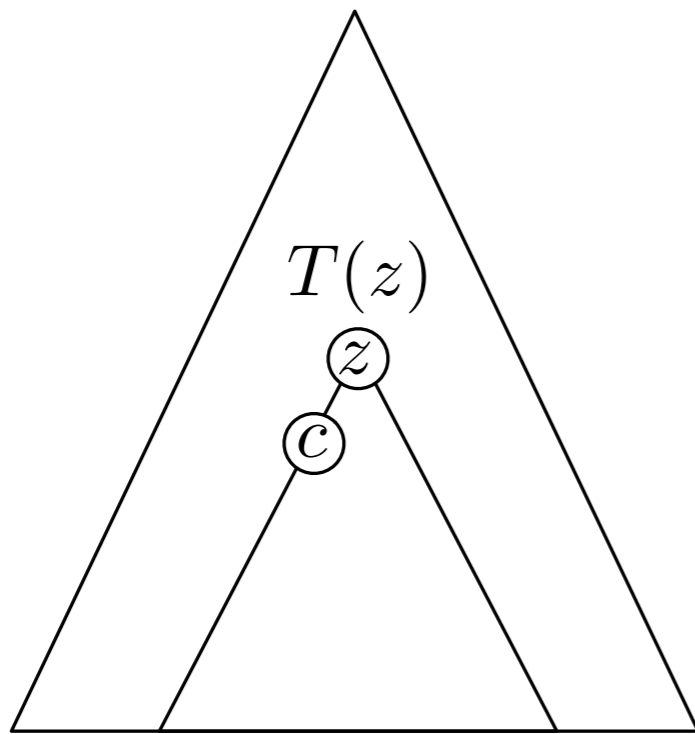
- How good is top tree compression?
- Worst case compression ratio.
- DAG vs. top tree compression.



- Identical clusters in T are represented by identical subtrees in top tree.
- Identical subtrees in top tree are merged in top DAG.
- \Rightarrow All clusters represented in top DAG are *distinct*.
- \Rightarrow Top tree contains at most $O(N/(\log_{\sigma} N)^{0.19})$ distinct clusters
- \Rightarrow Theorem: Top DAG has size at most $O(N/(\log_{\sigma} N)^{0.19})$

DAG vs. Top DAG

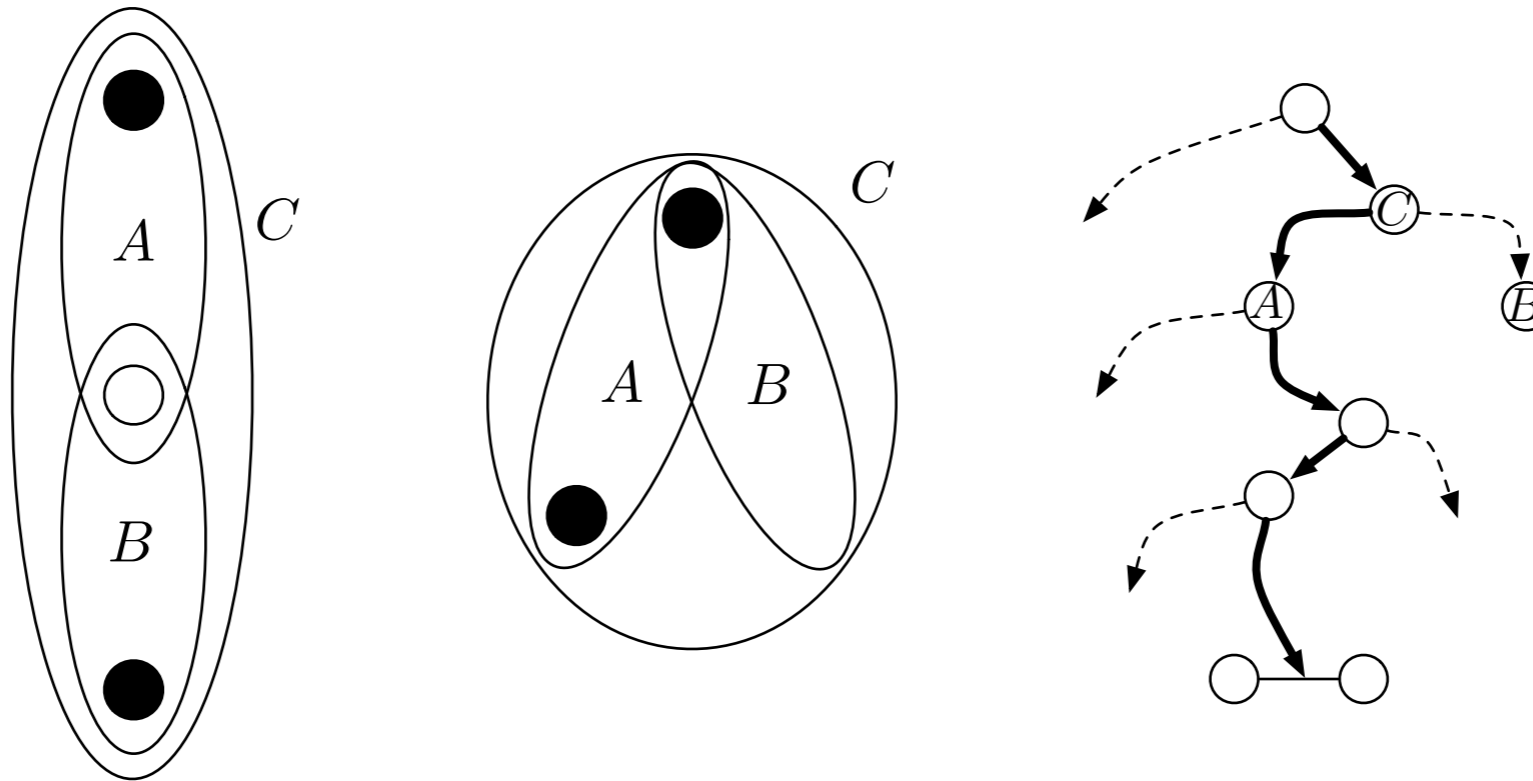
- How good is top DAG compression vs. DAG compression?



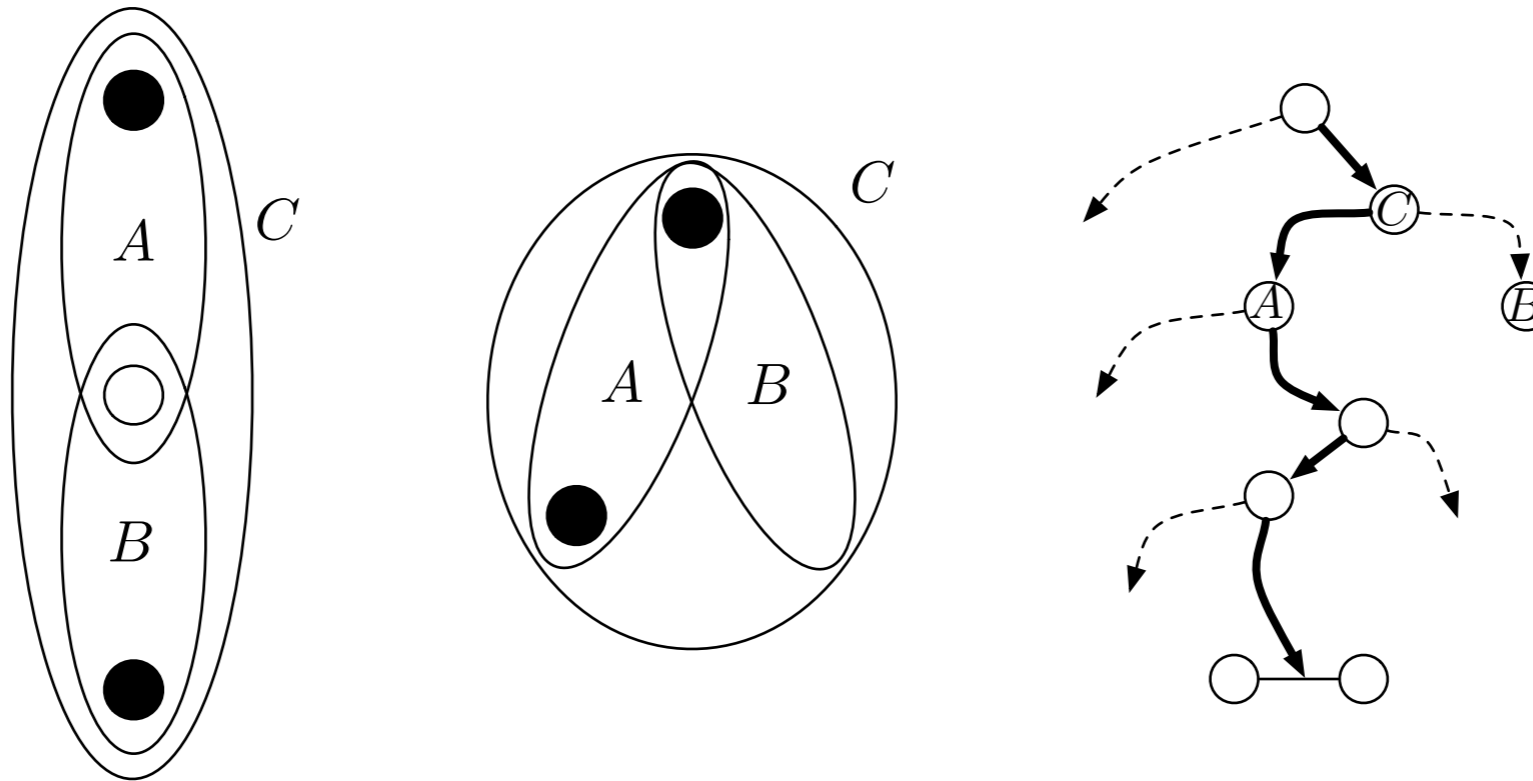
- Consider any subtree $T(z)$ in T . Suppose z has left child c .
- $T(z)$ is represented by a set S of $O(\log N)$ clusters in top tree (a subset of *off-path* clusters from the smallest cluster U containing $T(z)$ to (z,c)).
- Let $T(z')$ be a subtree repeat of $T(z)$. Then $T(z')$ is represented by a set S' of clusters in top tree identical to S .
- In DAG $T(z')$ is compressed to a single edge. In top DAG $T(z')$ each of the $O(\log N)$ clusters in S' are compressed to an edge.
- \Rightarrow Theorem: The top DAG has size $O(D \cdot \log N)$

Compressed Navigation

- Identify nodes in T by preorder number.
- Theorem: Using $O(n)$ space we can support the following operations in $O(\log N)$ time:
 - $\text{Access}(x)$: Return the label associated with node x .
 - $\text{Decompress}(x)$: Return the tree $T(x)$.
 - $\text{Parent}(x)$: Return the parent of node x .
 - $\text{Depth}(x)$: Return the depth of node x .
 - $\text{Height}(x)$: Return the height of node x .
 - $\text{Size}(x)$: Return the number of nodes in $T(x)$.
 - $\text{FirstChild}(x)$: Return the first child of x .
 - $\text{NextSibling}(x)$: Return the sibling immediately to the right of x .
 - $\text{LevelAncestor}(x, i)$: Return the ancestor of x whose distance from x is i .
 - $\text{NCA}(x, y)$: Return the nearest common ancestor of the nodes x and y .

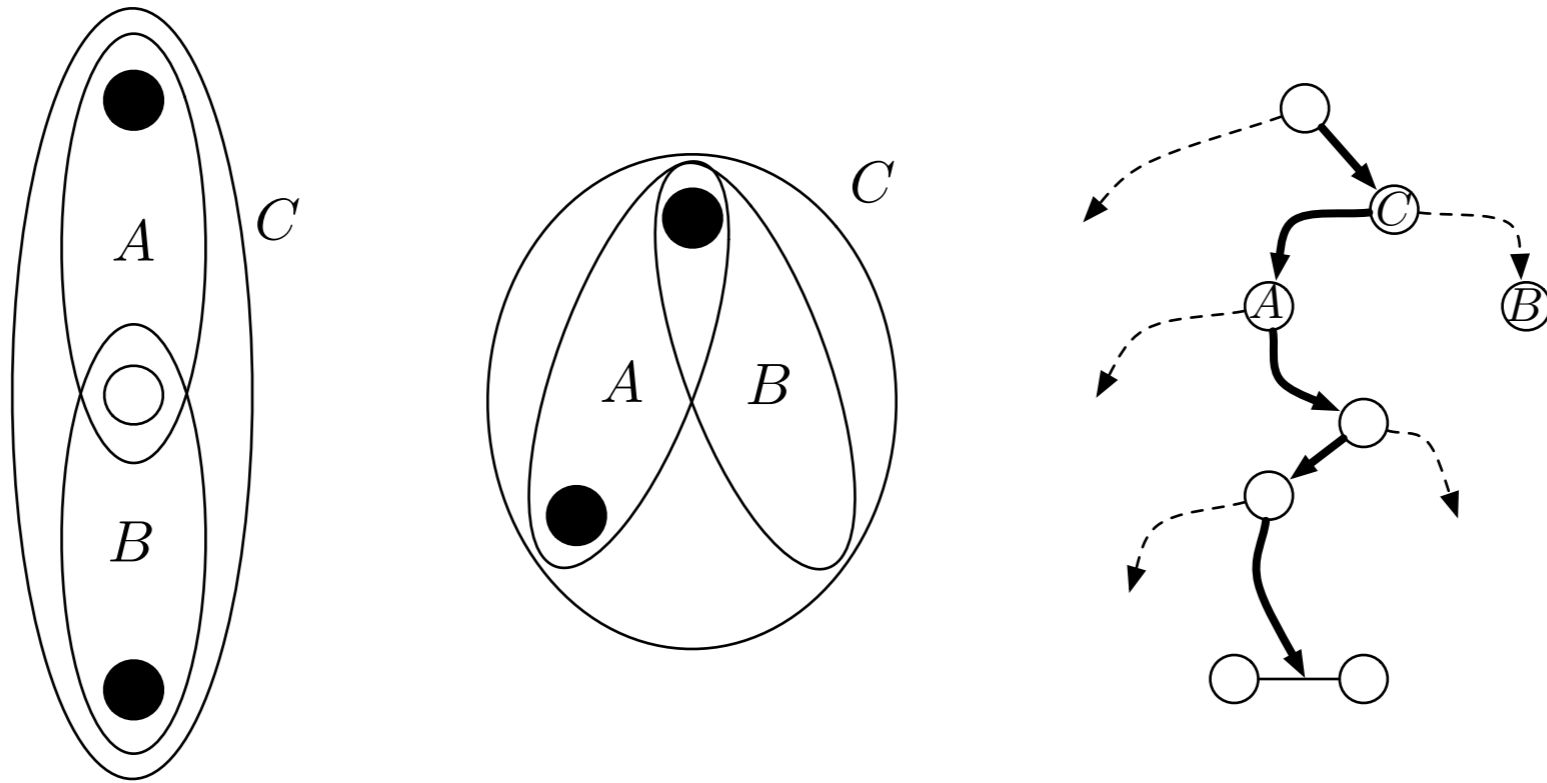


- Store $O(1)$ space information in each node of top DAG (type of merge, height of cluster, size of cluster, ...)
- Implement operations with top down and bottom up recursive searches in top DAG.
- Identify nodes by maintaining *local preorder number* during searches.
- Use constant time in each node $\Rightarrow O(\log N)$ time for operation.



- $NCA(x,y)$:

- Top down search for x and y to find smallest cluster C containing x and y .
- Retrieve local preorder number for $NCA(x,y)$ in C .
- Bottom up search to map local preorder number to global preorder number in T .



- $\text{Size}(x)$

- Top down search for x to find set of off-path cluster representing $T(x)$.
- Return sum of sizes of these cluster.

Summary and Open Problems

- Top tree compression
 - DAG compression of top tree
 - Compression ratio at least $(\log_{\sigma} N)^{0.19}$ and never more than a $\log N$ factor larger than DAG compression
 - Navigation in $O(\log N)$ time.
- Open problems
 - Improve $(\log_{\sigma} N)^{0.19}$ worst case compression ratio for top DAG compression.
 - Compressed pattern matching for trees compressed with repetitions.
 - Practical implementations.