Compressed Representations of Set Collections

Luca Lombardo

MSc Thesis - Extended Abstract

Succinct data structures represent combinatorial objects close to their information-theoretic minimum while supporting queries directly on the compressed form [1, Ch. 1]. We consider the problem of representing a collection $\mathscr{S} = \{S_1, \ldots, S_m\}$ of finite subsets of a totally ordered universe $\mathscr{U}$ of size u, supportingthe following operations:

- `membership`(x, S): given $x \in \mathscr{U}$, determine whether $x \in S$;
- `access`(i, S): given $i \in [1..|S|]$, return the i-th smallest element of S;
- `rank`(x, S): given $x \in \mathscr{U}$, return the number of elements of S that are $\leq x$;
- `predecessor/successor`(x, S): given $x \in \mathscr{U}$, find the largest element of S that is $\leq x$ and the smallest element of S that is $\geq x$, respectively.

These operations are the fundamental primitives for higher-level tasks such as union and intersection of sets in $\mathscr{S}$. A standard method for measuring the space needed to encode $\mathscr{S}$ is to treat each set S_i as a bitvector of length u indicating which elements of $\mathscr{U}$ belong to S_i. We call this cost independent encoding, and write it as $L_{\text{ind}}(\mathscr{S}) = \sum_i \lg \binom{u}{|S_i|}$ bits, the sum of the information content of each set when encoded independently. Using standard techniques (see [2]) we can store $\mathscr{S}$ in $L_{\text{ind}}(\mathscr{S}) + O(n + m \lg n)$ bits, where $n = \sum |S_i|$, supporting accessing any i-th element of any set in constant time.

Compressibility measures beyond the standard argument of above for set collections are recent: Alanko et al. [3] defined a containment entropy, and Gagie, He, and Navarro [4] defined insertion and symdiff compressibility, achieving compressed representations with logarithmic query times via the tree extraction framework of He, Munro, and Zhou [5, 6]. Both approaches, however, build hierarchies exclusively from the original sets, without introducing synthetic nodes such as $S_i \cup S_j$. Alanko et al. note that augmentation with new sets may be not computable in polynomial time; Gagie et al. leave optimal augmentation explicitly open.

We take a different approach: rather than encoding one original set in terms of another, we introduce *synthetic* nodes, sets not in $\mathscr{S}$, to serve as common ancestors of incomparable pairs. We present Set-Union Matching (SUM), a polynomial-time greedy algorithm that constructs hierarchies augmented with union nodes, guaranteeing total encoding cost always below the independent encoding cost $L_{\text{ind}}(\mathscr{S})$. The resulting hierarchy satisfies a pure-containment property that allows us to leverage the tree extraction framework of [5, 6], supporting the above operations with logarithmic query times.

Existing Frameworks for Compressing Set Collections

If $S_i \subset S_j$, then S_i can be described using $\lg\binom{|S_j|}{|S_i|}$ bits by indicating which elements of S_j belong to S_i. Alanko, Bille, Gørtz, Navarro, and Puglisi [3] exploit this observation by organizing $\mathscr{S}$ into a hierarchy rooted at $\mathscr{U}$: the parent $p(S_i)$ of each set is a smallest proper superset of S_i in $\mathscr{S}$ (or $\mathscr{U}$ if none exists), and S_i is stored as a sparse bitvector relative to $p(S_i)$. The resulting space is governed by the *containment entropy*

$$L(\mathscr{S}) = \sum_{i=1}^{m} \lg\binom{|p(S_i)|}{|S_i|},$$

which satisfies $L(\mathscr{S}) \le L_{\text{ind}}(\mathscr{S})$ because $|p(S_i)| \le u$ for every i. A path-compression technique guarantees depth $O(\lg(u/|S_i|))$ for each set without asymptotically affecting the space, yielding retrieval, predecessor, and successor queries in $O(\lg(u/|S|))$ time within $L(\mathscr{S}) + O(N + m\lg N)$ bits. They note that introducing sets not in $\mathscr{S}$, such as $S_i \cap S_j$, into the hierarchy could yield less space, but that finding the optimal augmentation may not be computable in polynomial time.

The containment entropy encodes each set relative to a superset. Gagie, He, and Navarro [4] take the dual perspective: each set S is instead described by specifying the elements it adds to its largest strict subset $p'(S)$ in $\mathscr{S}$ (or $\emptyset$ if none exists).

Definition 1. The insertion compressibility of $\mathscr{S}$ is $\mathscr{I}(\mathscr{S}) = \sum_{S\in\mathscr{S}} |S \setminus p'(S)|$.

An insertion graph on $\mathscr{S} \cup \{\emptyset\}$, with an edge from each S to $p'(S)$ of weight $|S \setminus p'(S)|$, forms a tree rooted at $\emptyset$ of total weight $\mathscr{I}(\mathscr{S})$.

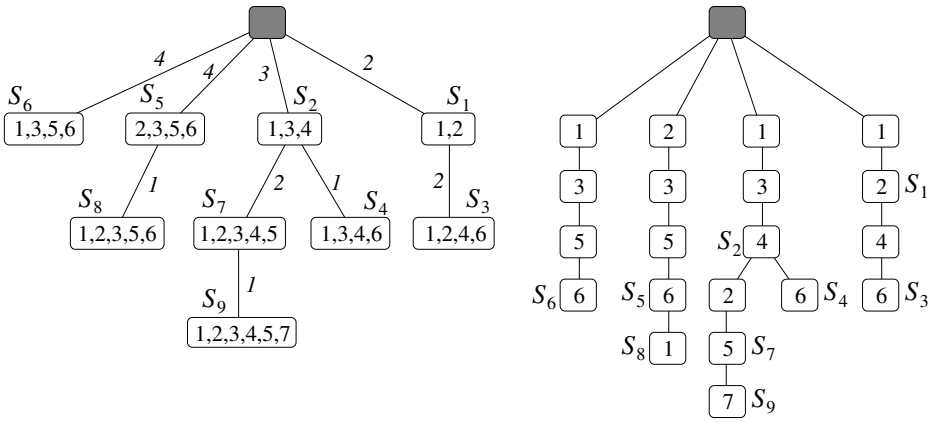

Figure 1: From [4]. On the left, an insertion graph for sets $\mathscr{S} = \{S_1, \ldots, S_9\}$, with the weights written as slanted values on the edges. It holds $\mathscr{I}(\mathscr{S}) = 20$. On the right, a corresponding insertion tree with 21 nodes. We write S_i besides its corresponding node $v(S_i)$.

To obtain efficient query support, Gagie et al. convert this graph into an insertion tree: the edge between $v(p'(S))$ and $v(S)$ is expanded into a chain of $|S \setminus p'(S)|$ edges, each labeled with a distinct element of $S \setminus p'(S)$, in any order (Figure 1). The resulting tree has $1 + \mathscr{I}(\mathscr{S})$ nodes, all labels from $[1..u]$, and no deletion marks. Since the content of

S is precisely the set of labels on the path from $v(\emptyset)$ to $v(S)$, applying tree extraction yields:

Theorem 2 (Gagie et al. [4])**.** *The insertion tree occupies $O(\mathscr{I}(\mathscr{S}))$ words. Membership is answered by* $\texttt{parent}_x(v(S)) \neq \perp$ *in $O(\log\log_\omega u)$ time; access is a path selection query in $O(\log_\omega u)$ time; rank is a path counting query with interval $[1,x]$ in $O(\log_\omega u)$ time; predecessor and successor combine rank and access in $O(\log_\omega u)$ time.*

Tree extraction, due to He, Munro, and Zhou [5, 6], represents a labeled ordinal tree in $O(|\mathscr{T}|)$ words and supports, for any node v and label α: $\texttt{parent}_\alpha(v)$ (lowest ancestor labeled α) in $O(\log\log_\omega u)$ time, and path counting (number of ancestors labeled in a range $[a,b]$) and path selection (i-th smallest ancestor label) queries in $O(\log_\omega u)$ time.

Insertion compressibility can only represent a set from a subset of it. To exploit the proximity of sets that share no containment relationship, Gagie et al. [4] define a stronger measure.

Definition 3. A symdiff graph on $\mathscr{S}$ is a weighted directed graph on $\mathscr{S} \cup \{\emptyset, \mathscr{U}\}$, where each $S \in \mathscr{S}$ has exactly one outgoing edge to some $p(S)$, of weight $|S \triangle p(S)|$, and every S reaches $\emptyset$ or $\mathscr{U}$. The symdiff compressibility $\Delta(\mathscr{S})$ is the minimum weight of such a graph.

Since any insertion tree is a valid symdiff graph, $\Delta(\mathscr{S}) \leq \mathscr{I}(\mathscr{S})$. The minimum-weight symdiff graph can be computed as the minimum spanning tree (MST) over the complete graph with nodes $\mathscr{S} \cup \{\emptyset, \mathscr{U}\}$ and edge weights $|S_i \triangle S_j|$, setting $w(\emptyset, \mathscr{U}) = 0$; removing the zero-cost edge yields two rooted trees, one at $\emptyset$ and one at $\mathscr{U}$, see Figure 2.

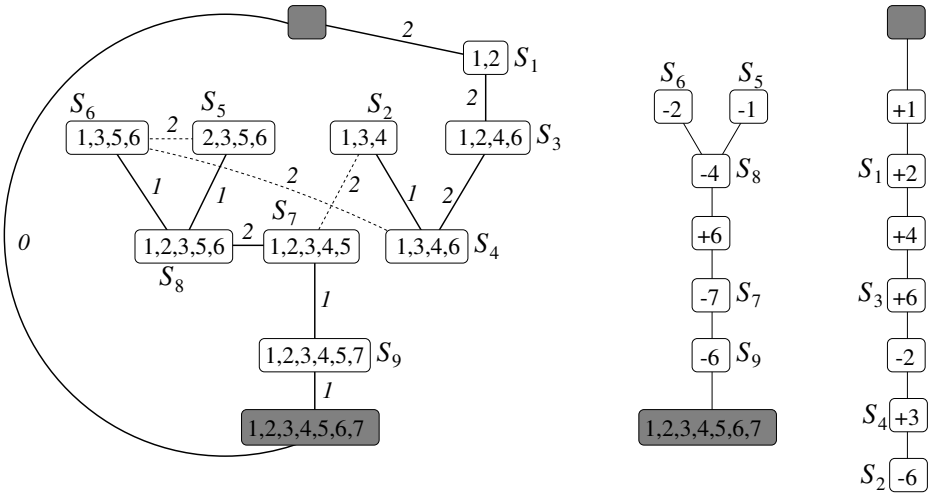

Figure 2: From [4]. On the left, a symdiff graph of minimum weight between the same sets of Figure 1, yielding $\Delta(\mathscr{S}) = 13$. To build it, it is sufficient to consider the full graph edges with weights up to $\ell = 2$. Among these edges, those not belonging to the MST (i.e., the symdiff graph of minimum weight) are dashed. On the right, the indel trees, rooted at $\mathscr{U}$ (upside down) and at $\emptyset$, that represent $\mathscr{S}$ in space $O(\Delta(\mathscr{S}))$. We write $+x$ and $-x$ to indicate elements x to add or to delete, respectively, from the parent node.

The difference from insertion trees is that edges in indel trees carry both insertion marks $+x$ and deletion marks $-x$ with respect to the parent set, so labels range over $[-u..u]$ and the indel trees have $\Delta(\mathscr{S})+2$ nodes in total. On insertion trees, access reduces to a path selection query on a single wavelet-like [7] hierarchy in $O(\log_\omega u)$ time. On indel trees, the presence of deletion marks means that a label x on the root-to-$v(S)$ path does not unambiguously indicate membership. Gagie et al. handle this by maintaining two separate hierarchical partitions, $\mathscr{T}^+$ for insertion marks and $\mathscr{T}^-$ for deletion marks, and descending both simultaneously: at each level, the number of elements of $[a..\lfloor(a+b)/2\rfloor]$ surviving in S is computed as $s^+ - s^-$, where s^+ and s^- count the insertions and deletions observed so far. Each level takes $O(1)$ time, but the binary partition forces $O(\log u)$ levels, yielding:

Theorem 4 (Gagie et al. [4])**.** *The indel tree representation occupies $O(\Delta(\mathscr{S}))$ words. Membership takes $O(\log\log_\omega u)$ time; rank takes $O(\log_\omega u)$ time; access, predecessor, and successor take $O(\log u)$ time.*

The degradation of access from $O(\log_\omega u)$ to $O(\log u)$ is a direct consequence of the coordinated two-hierarchy descent. Whether access can be improved to $O(\log_\omega u)$ in the symdiff setting is left open by [4].

Lower Bounds for Set Collections Any encoding of $\mathscr{S}$ must use enough bits to distinguish it from every other collection of m sets over $\mathscr{U}$. The Venn diagram of m sets partitions $\mathscr{U}$ into at most 2^m-1 non-empty regions: the membership pattern of an element x is $\sigma(x)=\{i\in[m] : x\in S_i\}$, a non-empty subset of $[m]$. Since the pattern of each element is independent, the number of distinct collections is $(2^m-1)^u$, giving a worst-case lower bound of $u\lg(2^m-1)$ bits.

A specific collection, however, may populate far fewer than 2^m-1 regions. The non-empty regions are the *atoms* $A_\tau=\{x\in\mathscr{U} : \sigma(x)=\tau\}$: groups of elements that have identical membership across all m sets. If only k distinct patterns are realized, any encoding must distinguish the $\binom{u}{|A_{\tau_1}|,\ldots,|A_{\tau_k}|}$ assignments of elements to atoms.

Proposition 5. *Let $\tau_1,\ldots,\tau_k$ be the distinct patterns realized by $\mathscr{S}$. Then*

$$L^*(\mathscr{S}) \;=\; \lg\binom{u}{|A_{\tau_1}|,\ldots,|A_{\tau_k}|}$$

is a lower bound on the space required by any encoding of $\mathscr{S}$.

The chain $L^*(\mathscr{S})\le L_{\mathrm{ind}}(\mathscr{S})\le u\lg(2^m-1)$ holds universally. When $k\ll 2^m-1$, the gap between L^* and L_{ind} can be substantial; characterizing how close we can approach L^* is the central open problem that motivates our work.

Set-Union Matching

The containment entropy of [3] and the symdiff compressibility of [4] share a structural constraint: the nodes of their hierarchies correspond to the sets in $\mathscr{S}$ themselves (together with $\emptyset$ and, for the symdiff case, $\mathscr{U}$). This constraint has different consequences

in the two frameworks. In the containment setting, if $A \subset B$, then A is encoded as a subset of B at a cost governed by $|B|$ rather than u; but if neither $A \subseteq B$ nor $B \subseteq A$, neither can serve as the parent of the other, and the encoding cost of both depends on the size of their closest superset in $\mathscr{S}$, which may be $\mathscr{U}$ itself. In the symdiff setting, incomparable pairs are handled via symmetric differences, but the resulting indel trees carry both insertion and deletion marks, forcing the coordinated two-hierarchy descent that degrades access to $O(\log u)$ (Theorem 4). Alanko et al. [3] observe that augmenting a hierarchy with sets not in $\mathscr{S}$ may be intractable; Gagie et al. [4] leave the complexity of optimal augmentation as an explicit open problem.

To address the limitations of both frameworks, we introduce *synthetic union* nodes to serve as common ancestors of incomparable pairs. The simplest and most natural such construction is the union.

Consider two sets $A, B \subseteq \mathscr{U}$, with neither $A \subseteq B$ nor $B \subseteq A$ and $A \cap B \neq \emptyset$. Both A and B can therefore be encoded as subsets of $M = A \cup B$ rather than of the full universe $\mathscr{U}$: when $|M| \ll u$, the binomial coefficient $\lg\binom{|M|}{|A|}$ is dramatically smaller than $\lg\binom{u}{|A|}$.

Encoding A and B *jointly* given M is cheaper still. A bitvector of length $|M|$ marks with 1 the $k = |A \cap B|$ elements of the intersection; its $|M| - k$ zeros correspond to the symmetric difference $A \triangle B$. A second bitvector, of length $l + r$ where $l = |A \setminus B|$ and $r = |B \setminus A|$, records which of these elements belong to $A \setminus B$ (marked 1) and which to $B \setminus A$ (the complementary zeros). The first bitvector encodes one of $\binom{|M|}{k}$ choices; the second, one of $\binom{l+r}{l}$. Their product $\binom{|M|}{k}\binom{l+r}{l} = \binom{|M|}{k,l,r}$ is the multinomial coefficient, and $\lg\binom{|M|}{k,l,r}$ is the information content of the merge.

Definition 6. Given $A, B \subseteq \mathscr{U}$, write $M = A \cup B$, $k = |A \cap B|$, $l = |A \setminus B|$, $r = |B \setminus A|$, so that $|M| = k + l + r$. The *merge cost* is

$$w(A,B) \;=\; \lg\binom{|M|}{k,\,l,\,r} \;+\; c(|M|),$$

where $c(|M|) = \lceil \lg(|M|+1) \rceil + \lceil \lg(|M|-k+1) \rceil$ bits encode k and l explicitly (from which $r = |M| - k - l$ follows) to make the representation self-delimiting. When $A \cap B = \emptyset$, the parameter $k = 0$ is implicit, the multinomial reduces to $\lg\binom{l+r}{l}$, and the overhead to a single $\lceil \lg(|M|+1) \rceil$ term.

The cost $w(A,B)$ is conditional: it measures the bits to recover A and B given $M = A \cup B$, but says nothing about how M itself is specified. If no other information is available, M must be encoded as a subset of $\mathscr{U}$ at a cost of $\lg\binom{u}{|M|}$ bits, making the total cost of representing the pair $\lg\binom{u}{|M|} + w(A,B)$. But if M has itself been obtained as the union of two other sets and is already available from that context, its encoding cost has already been paid elsewhere, and only the conditional term $w(A,B)$ remains. The total cost of representing A and B therefore depends on whether M must be specified from scratch or is inherited from a prior merge.

This observation suggests a recursive strategy for an entire collection $\mathscr{S} = \{S_1, \ldots, S_m\}$. Suppose we merge A and B into $M = A \cup B$, and separately merge C and D into $M' = C \cup D$, paying $w(A,B)$ and $w(C,D)$ respectively. We can then merge M and M' into $M'' = M \cup M'$ at a further cost of $w(M,M')$; since both M and M' are now recovered

from M'', only M'' itself must be encoded against $\mathcal{U}$. The encoding chains: M'' is specified as a subset of $\mathcal{U}$, the pair (M, M') is recovered from M'', and finally (A, B) and (C, D) from their respective parents. Each merge transfers cost from the global level, encoding against $\mathcal{U}$, to a local level, encoding against a smaller union.

Iterating this process yields a binary forest $\mathcal{F}$ whose m leaves are the original sets and whose internal nodes are synthetic unions: each internal node v has two children A_v, B_v and represents $M_v = A_v \cup B_v$. A root R of the forest has no parent and must be encoded against $\mathcal{U}$ at a cost of $\lg\binom{u}{|R|}$ bits; an internal node v encodes the decomposition of M_v into A_v and B_v at the conditional cost $w(A_v, B_v)$. The total cost is

$$\Phi(\mathcal{F}) = \underbrace{\sum_{R \in \text{roots}(\mathcal{F})} \lg\binom{u}{|R|}}_{\text{support cost}} + \underbrace{\sum_{v \in \text{internal}(\mathcal{F})} w(A_v, B_v)}_{\text{structural cost}}. \tag{1}$$

Every set in $\mathcal{S}$ is recoverable by decoding each root against $\mathcal{U}$ and applying the conditional decompositions along the path from root to leaf.

When no merges are performed, each S_i is itself a root and $\Phi(\mathcal{F}_0) = \sum_i \lg\binom{u}{|S_i|} = L_{\text{ind}}(\mathcal{S})$. The question is therefore: given $\mathcal{S}$, which pairs should be merged, and in which order?

Finding the forest that minimizes Φ is a combinatorial optimization over all binary hierarchies on $\mathcal{S}$; the complexity of this problem is unknown, and both [3] and [4] note that optimal augmentation with synthetic nodes may be intractable. We propose a polynomial-time greedy heuristic.

Definition 7. The *Set-Union Matching* (SUM) algorithm initializes a forest $\mathcal{F}_0$ of m isolated roots, one per $S_i \in \mathcal{S}$. At each level $t \geq 0$, it evaluates $w(R_i, R_j)$ for every pair of current roots and extracts a greedy minimum-weight matching: repeatedly select the pair with smallest merge cost and remove both endpoints. Each matched pair (A, B) is replaced by a single new root $M = A \cup B$ with children A and B. Since each level reduces the number of roots by at least a factor of two, the procedure terminates in at most $\lceil \lg m \rceil$ levels.

The greedy matching at each level is a local decision, and the global cost $\Phi(\mathcal{F}_t)$ need not decrease monotonically.

Proposition 8. *There exist collections $\mathcal{S}$ for which $\Phi(\mathcal{F}_{t+1}) > \Phi(\mathcal{F}_t)$ at some level t.*

Proof. Let $\mathcal{S}$ contain four pairwise-disjoint sets of size $u/4$. At level 0, two merges produce two roots of size $u/2$; each merge adds $\lg\binom{u/2}{u/4} \approx u/2$ bits of structural cost while saving $2\lg\binom{u}{u/4} - \lg\binom{u}{u/2}$ bits of support cost, a net decrease. At level 1, merging the two roots of size $u/2$ produces $M = \mathcal{U}$ with $\lg\binom{u}{u} = 0$ support cost, but adds $\lg\binom{u}{u/2} \approx u$ bits of structural cost. The total after level 1 exceeds the total after level 0. □

Merging to completion can therefore overshoot the configuration of minimum cost. We address this with an *optimal-forest* variant: since SUM proceeds level by level, we can record $\Phi(\mathcal{F}_t)$ after each level and compare it with the best value seen so far. When the algorithm terminates, we return not the final forest but the snapshot $\mathcal{F}_{t^*}$ that achieved the smallest Φ across all levels.

Theorem 9. *For any $\mathscr{S}$ over a universe of size u, SUM with optimal-forest stopping produces a forest $\mathscr{F}_{t^*}$ satisfying*

$$L_{\mathrm{SUM}}(\mathscr{S}) \;=\; \Phi(\mathscr{F}_{t^*}) \;\le\; L_{\mathrm{ind}}(\mathscr{S}).$$

Proof. The initial forest $\mathscr{F}_0$ has cost $L_{\mathrm{ind}}(\mathscr{S})$ and is one of the snapshots over which the minimum is taken; hence $\Phi(\mathscr{F}_{t^*}) \le \Phi(\mathscr{F}_0) = L_{\mathrm{ind}}(\mathscr{S})$. □

The guarantee $L_{\mathrm{SUM}}(\mathscr{S}) \le L_{\mathrm{ind}}(\mathscr{S})$ is universal but may be far from tight: for collections with rich structure, the atom-based lower bound $L^*(\mathscr{S})$ of Proposition 5 can be substantially smaller than $L_{\mathrm{ind}}(\mathscr{S})$, and characterizing the gap between L_{SUM} and L^* across different collection families remains open. The computational cost is dominated by level 0, where $O(m^2)$ pairwise merge costs must be evaluated, each in $O(|A|+|B|)$ time; over $\lceil \lg m \rceil$ levels the worst case is $O(m^2 N \lg m)$.

The cost $\Phi(\mathscr{F}_{t^*})$ characterizes the information content of the SUM hierarchy but does not by itself yield a queryable data structure. To support operations on the sets in $\mathscr{S}$, we encode the hierarchy as a labeled ordinal tree and apply the tree extraction framework of [5, 6].

Proposition 10. *Every parent-child edge in the SUM hierarchy is a containment from parent to child: for every internal node v with children A_v, B_v and union $M_v = A_v \cup B_v$, both $A_v \subseteq M_v$ and $B_v \subseteq M_v$, with equality if and only if one child already contains the other.*

This is immediate from the definition of M_v, but the structural consequence is significant. Every label on a root-to-leaf path represents an element *removed* from the parent. Once an element is absent from a node, it is absent from all its descendants, so no label can appear twice on the same path. The labeled tree therefore carries labels from $[1..u]$ only, with no insertion marks, and the content of each set S is determined by the labels on its root-to-$v(S)$ path. We call the resulting structure a *deletion tree*, in contrast with the insertion tree of Theorem 2.

When containment relationships exist ($S_i \subset S_j$), SUM captures the same savings as the containment entropy of [3] via trivial merges in which $|A \setminus B| = 0$: the synthetic node $M = A \cup B = B$ coincides with the larger set, and the dominant term of the merge cost reduces to $\lg \binom{|B|}{|A|}$, the containment entropy contribution for the pair; the self-delimiting overhead $c(|B|) = O(\lg |B|)$ bits remains. The advantage of SUM lies in incomparable pairs, where the containment hierarchy cannot place either set as the parent of the other and may be forced to encode both against $\mathscr{U}$.

Consider a tree in the SUM forest rooted at R. We expand each edge (M_v, A_v) into a chain of $|M_v \setminus A_v| = |B_v \setminus A_v|$ nodes, each labeled with a distinct element of $M_v \setminus A_v$; symmetrically for (M_v, B_v). The total number of chain nodes emanating from v is $|A_v \triangle B_v|$, and all labels lie in $[1..u]$. The resulting labeled ordinal tree has $1 + \sum_v |A_v \triangle B_v|$ nodes. In the insertion tree of [4], the root is $\emptyset$ and the labels on the root-to-$v(S)$ path accumulate to give $S = \{\text{labels on path}\}$. In the SUM deletion tree, the root is R and the content of $v(S)$ is $R \setminus \{\text{labels on path}\}$: the path records elements removed from R, not elements added to $\emptyset$.

Since all labels lie in $[1..u]$ with no sign, the deletion tree avoids the coordinated two-hierarchy descent that the indel framework of [4] requires for sets encoded via both insertions and deletions. We now describe how each operation is supported using [5] and [6].

For membership, $x \in S$ if and only if two conditions hold: x belongs to the root set R, and no ancestor of $v(S)$ carries label x (meaning x was never removed along the path from R to S). The first condition is answered in $O(1)$ time by a rank structure on R. The second reduces to $\texttt{parent}_x(v(S)) = \perp$ in the labeled tree framework of [5], which runs in $O(\log\log_\omega u)$ time.

For rank, the number of elements of S that are at most x equals the number of elements of R that are at most x, minus the number of deleted elements in $[1..x]$ along the root-to-$v(S)$ path. The first quantity is $|R \cap [1..x]|$, computed in $O(1)$ via the same rank structure on R; this structure occupies $\lg\binom{u}{|R|}$ bits, already accounted for in the support cost of the root (Equation 1). The second quantity is a path counting query $\texttt{pathcount}_{[1,x]}(v(S))$ in the hierarchical partition of [6], answered in $O(\log_\omega u)$ time.

Access to the ith smallest element of S reduces to finding the ith smallest element of R that is *not* a label on the root-to-$v(S)$ path. In the insertion tree of [4], access reduces to path selection (finding the ith smallest label among the ancestors), and the sublogarithmic decomposition of [6] solves this in $O(\log_\omega u)$ time. In the deletion tree, however, the required operation is complementary: selecting the ith smallest element of R that is absent from the path. We solve this using the binary wavelet-like partition of [6], which partitions the universe via successive halving. At each level, the range $[a..b]$ splits into $[a..\lfloor(a+b)/2\rfloor]$ and $[\lfloor(a+b)/2\rfloor+1..b]$, and each node of the binary tree $T_{a,b}$ receives label 0 or 1 according to whether its original label falls in the left or right half of $[a..b]$. The count of surviving elements in the left half is then $|R \cap [a..\lfloor(a+b)/2\rfloor]|$ minus $\texttt{rank}_0(v)$ in $T_{a,b}$, where $\texttt{rank}_0$ counts the deletion labels that fall in the left half; since $T_{a,b}$ has alphabet $\{0,1\}$, this is computable in $O(1)$ time. The binary partition has $O(\log u)$ levels, so access runs in $O(\log u)$ time. In the indel framework of [4], two separate hierarchical partitions (T^+ for insertion marks and T^- for deletion marks) must be descended simultaneously at each level. The deletion tree eliminates T^+ entirely, requiring only a single partition of $[1..u]$; but at each level the count of surviving elements still combines the partition with an external rank query on R, rather than reading path labels directly, so the complementary nature of the selection persists. Predecessor and successor combine rank and access, so they also run in $O(\log u)$ time.

Theorem 11. *On a word RAM with ω-bit words, the SUM hierarchy for $\mathscr{S}$ can be represented via tree extraction [5, 6] in $O(\sum_v |A_v \triangle B_v|)$ words per tree, supporting membership in $O(\log\log_\omega u)$ time, rank in $O(\log_\omega u)$ time, and access, predecessor, successor in $O(\log u)$ time.*

The space in Theorem 11 is measured in words. The information-theoretic cost $L_{\mathrm{SUM}} = \Phi(\mathscr{F}_{t^*})$ accounts for each merge via $\lg\binom{|M_v|}{k_v,l_v,r_v}$, which depends on the *local* universe $|M_v|$ rather than on u. The tree extraction representation, on the other hand, pays $O(\sum_v |A_v \triangle B_v| \cdot \lg u)$ bits, since each chain node carries a label from the global alphabet $[1..u]$. When $|M_v| \ll u$, the gap is substantial: for bounded $|M_v|$ and growing

u, the multinomial contributes $O(\lg|M_v|)$ bits per merge, while tree extraction pays $\Theta(|A_v \triangle B_v| \cdot \lg u)$ bits. This discrepancy is inherent in tree extraction, whose labels reside in a global alphabet.

The SUM hierarchy also admits an alternative representation. The augmented collection $\mathscr{S}_{\text{aug}} = \mathscr{S} \cup \{M_v : v \text{ internal}\}$ includes the original sets together with the synthetic union nodes. This collection can be fed into the indel tree framework of [4], yielding a representation in $O(\Delta(\mathscr{S}_{\text{aug}}))$ words that supports membership in $O(\log\log_\omega u)$, rank in $O(\log_\omega u)$, and access in $O(\log u)$, for all sets in $\mathscr{S}$. The relationship between $\Delta(\mathscr{S}_{\text{aug}})$ and $\Delta(\mathscr{S})$ is instance-dependent: the synthetic vertices introduce short-range edges in the complete graph (since $|M_v \triangle A_v| = |B_v \setminus A_v| \le |A_v \triangle B_v|$), but the MST of $\mathscr{S}_{\text{aug}} \cup \{\emptyset, \mathscr{U}\}$ must also span the additional nodes, and the net effect can be an increase. For collections with high pairwise overlap, however, $\Delta(\mathscr{S}_{\text{aug}})$ can be substantially smaller than $\Delta(\mathscr{S})$.

Open Questions & Further Work

The greedy matching in SUM is a heuristic; the complexity of finding the augmentation minimizing total encoding cost is unknown. The gap between $L_{\text{SUM}}(\mathscr{S})$ and the atom-based lower bound $L^*(\mathscr{S})$ remains uncharacterized. On the computational side, the $O(m^2)$ pairwise comparisons per level become prohibitive for large collections; pruning the candidate pairs via locality-sensitive hashing [8, 9] or replacing the symmetric difference with alternative proximity measures are natural directions to explore.

Whether the pure-containment structure of the SUM deletion tree admits an improved access bound of $O(\log_\omega u)$ is a natural question. The obstacle is that in an insertion tree, path selection directly yields the ith label in $O(\log_\omega u)$ time via the sublogarithmic decomposition of [6], while in a deletion tree, the complementary operation of selecting the ith *non*-label from the universe is not among the primitives currently supported by tree extraction. Gagie et al. [4] leave this open for pure-deletion trees more generally.

More broadly, the tension between bits and words suggests a further challenge: achieving space proportional to L_{SUM}, or to $\Delta(\mathscr{S}_{\text{aug}})$, while retaining the $O(\log_\omega u)$ access time that the absence of deletion marks grants in the pure-insertion setting. Such a representation would require either a variant of tree extraction that operates on local alphabets, or an entirely different query mechanism; neither is available with current techniques.

This extended abstract presents the theoretical analysis of the SUM algorithm; the implementation and experimental evaluation remain to be carried out. Several claims in this work, such as the potential advantage of $\Delta(\mathscr{S}_{\text{aug}})$ over $\Delta(\mathscr{S})$ for collections with high pairwise overlap, are supported by the analysis but await quantification on concrete instances. The final version of the thesis will include a full implementation of the SUM construction pipeline and a systematic experimental comparison against the frameworks of [3, 4] on real-world datasets, including web graph adjacency lists, inverted indexes, and colour sets from pangenomic de Bruijn graphs.

References

[1] G. Navarro. *Compact data structures: A practical approach.* Cambridge University Press, 2016.

[2] D. Okanohara and K. Sadakane. "Practical entropy-compressed rank/select dictionary". In: *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM. 2007, pp. 60–70.

[3] J. N. Alanko et al. "Compact data structures for collections of sets". In: *23rd Symposium on Experimental Algorithms*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2025, p. 6.

[4] T. Gagie, M. He, and G. Navarro. "Compressed Set Representations Based on Set Difference". In: *arXiv preprint arXiv:2601.23240* (2026).

[5] M. He, J. I. Munro, and G. Zhou. "A framework for succinct labeled ordinal trees over large alphabets". In: *Algorithmica* 70.4 (2014), pp. 696–717.

[6] M. He, J. I. Munro, and G. Zhou. "Data structures for path queries". In: *ACM Transactions on Algorithms* 12.4 (2016), 53:1–53:32.

[7] R. Grossi, A. Gupta, and J. S. Vitter. "High-order entropy-compressed text indexes". In: (2003).

[8] A. Z. Broder. "On the resemblance and containment of documents". In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE. 1997, pp. 21–29.

[9] M. S. Charikar. "Similarity estimation techniques from rounding algorithms". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. 2002, pp. 380–388.