Open Access and Peer Reviewed Journal ISSN 2394-2231 https://www.ijctjournal.org/

RESEARCH ARTICLE OPEN ACCESS

A JOIN-TREE SUFFICIENT CONDITION FOR THE NON-CANCELLING INTERSECTIONS CONJECTURE

MICHAEL S. YANG
Independent Researcher
yangofzeal@gmail.com

Abstract

The Non-Cancelling Intersections (NCI) Conjecture of Amarilli, Monet, and Suciu asks whether the union of a finite inclusion-incomparable family can always be constructed from exactly those intersections whose collected inclusion–exclusion coefficients are nonzero, using only disjoint union and subset complement. We prove a structural sufficient condition formulated in terms of an acyclic hypergraph representation. Let $\mathcal{N}$ be the family of non-cancelling intersections and let $\mathcal{B}$ be its inclusion-maximal members. If $\mathcal{B}$ admits a join tree satisfying the running-intersection property and every nonempty edge separator of that tree belongs to $\mathcal{N}$, then $\bigcup\mathcal{F}$ has an explicit left-linear construction over $\mathcal{N}$. Rooting the join tree, one subtracts the parent separator from each child bag and then adjoins the resulting residue by disjoint union. Running intersection guarantees that each residue is disjoint from everything assembled earlier. This yields a linear-size structural certificate and a direct polynomial-time verifier once the join tree is supplied. We compare the theorem with the known partial cases and strengthenings in the original NCI paper and with the recent failure of unrestricted left-linear witnesses. As supporting evidence only, a public implementation verified 50,000 independently generated theorem-eligible instances, including examples with eleven collected cancelling intersections. The theorem and verifier are independent of any private search acceleration.

Keywords: non-cancelling intersections, inclusion–exclusion, Möbius inversion, join trees, acyclic hypergraphs, running intersection, disjoint union, subset complement.

1 Introduction

The Non-Cancelling Intersections (NCI) Conjecture was introduced by Amarilli, Monet, and Suciu in connection with a question from probabilistic databases and knowledge compilation [1]. In its combinatorial form, it asks whether algebraic cancellation in inclusion–exclusion can always be mirrored by a constructive set expression. More precisely, after collecting equal intersections and discarding those whose total inclusion–exclusion coefficient is zero, can the original union still be assembled from the surviving intersections using only disjoint unions and subset complements?

The full conjecture remains open. The original paper proves the conjecture when there are no cancelling intersections and when there is exactly one cancelling intersection, develops an equivalent Boolean-lattice formulation, and proposes stronger variants, including a left-linear form [1]. The left-linear strengthening is now known to fail in general by work of Wilhelm [2]. Thus there is value in identifying natural structural classes on which left-linear witnesses are nevertheless forced.

This paper gives such a class. The relevant structure is acyclicity of the maximal non-cancelling intersections in the sense of a join tree. Join trees and the running-intersection property are standard descriptions of acyclic hypergraphs and database schemes [3]. The central observation here is that, if every edge separator in such a join tree is itself non-cancelling, then each child bag contributes a residue that is automatically disjoint from the union of all previously processed bags. The NCI expression is then obtained by a single rooted sweep of the tree.

The proof is constructive and independent of computation. The computational section is intentionally postponed until after the structural comparison and algorithmic consequences. Its role is only to stress-test the construction and provide an independently reproducible verifier.

Let

$$\mathcal{F} = \{S_1, \ldots, S_n\}$$

be a finite family of finite sets. Throughout, $\mathcal{F}$ is assumed pairwise inclusion-incomparable:

$$S_i \not\subseteq S_j, \qquad S_j \not\subseteq S_i \quad (i \neq j).$$

For every nonempty $I \subseteq [n]$, write

$$S_I = \bigcap_{i \in I} S_i.$$

When equal intersections are collected, define

$$\mu(X) = \sum_{\substack{\varnothing \neq I \subseteq [n] \\ S_I = X}} (-1)^{|I|+1}.$$

This is the collected inclusion–exclusion coefficient of X. It is the relevant Möbius coefficient on the intersection poset; see the classical theory of Möbius inversion [4] and the standard finite-poset treatment [5].

Definition 1 (Non-cancelling intersections)**.** The family of non-cancelling intersections is

$$\mathcal{N} = \{X : \mu(X) \neq 0\}.$$

Following Amarilli–Monet–Suciu, we use two partial set operations:

$$A \dot{\cup} B = A \cup B \quad \text{when } A \cap B = \varnothing,$$

and

$$A \dot{\setminus} B = A \setminus B \quad \text{when } B \subseteq A.$$

Let $\bullet(\mathcal{N})$ denote the sets constructible from leaves in $\mathcal{N}$ by legal applications of these two operations. The NCI Conjecture asserts

$$\bigcup \mathcal{F} \in \bullet(\mathcal{N})$$

for every finite pairwise inclusion-incomparable family $\mathcal{F}$ [1].

3 The Main Theorem

Let

$$\mathcal{B} = \{B_1, \ldots, B_m\}$$

be the inclusion-maximal members of $\mathcal{N}$.

Definition 2 (Join tree)**.** A tree T with vertex set $\mathcal{B}$ is a *join tree* if, for every ground element x, the set of bags containing x induces a connected subtree of T.

Equivalently, if $x \in B_i \cap B_j$, then x belongs to every bag on the unique B_i–B_j path. This is the running-intersection property.

Definition 3 (Separator-noncancelling join tree)**.** A join tree T is *separator-noncancelling* if, for every edge BB' of T,

$$B \cap B' = \varnothing \quad \text{or} \quad B \cap B' \in \mathcal{N}.$$

We first note why the maximal non-cancelling bags cover the desired union.

Lemma 1. *Every original set $S_i \in \mathcal{F}$ belongs to $\mathcal{N}$.*

satisfied $S_I = S_i$, then I would contain some $j \neq i$, giving

$$S_i = S_I \subseteq S_j,$$

contrary to inclusion-incomparability. Hence the singleton term is the unique contribution and $\mu(S_i) = 1$. □

Corollary 2. *The maximal members of $\mathcal{N}$ cover the target:*

$$\bigcup_{B \in \mathcal{B}} B = \bigcup_{S \in \mathcal{F}} S.$$

Proof. By Lemma 1, every S_i belongs to $\mathcal{N}$, hence is contained in a maximal member of the finite family $\mathcal{N}$. Conversely, every member of $\mathcal{N}$ is an intersection of members of $\mathcal{F}$ and is therefore contained in $\bigcup \mathcal{F}$. □

Theorem 3 (Join-Tree NCI Theorem)**.** *Let $\mathcal{F}$ be a finite pairwise inclusion-incomparable family, let $\mathcal{N}$ be its non-cancelling intersections, and let $\mathcal{B}$ be the maximal members of $\mathcal{N}$. If $\mathcal{B}$ admits a separator-noncancelling join tree, then*

$$\boxed{\bigcup \mathcal{F} \in \bullet(\mathcal{N}).}$$

Moreover, the construction can be chosen left-linear.

Proof. Root the join tree T at an arbitrary bag B_r. For each non-root bag B, let $P(B)$ be its parent and define

$$D_B = B \cap P(B), \qquad R_B = B \setminus D_B.$$

By the separator-noncancelling hypothesis, either $D_B = \varnothing$ or $D_B \in \mathcal{N}$. Thus R_B is legally constructible from elements of $\mathcal{N}$:

$$R_B = \begin{cases} B, & D_B = \varnothing, \\ B \mathbin{\dot{\setminus}} D_B, & D_B \neq \varnothing. \end{cases}$$

Process the rooted tree in any parent-before-child order. Start with

$$X_0 = B_r.$$

Assume that the already processed bags form a connected rooted subtree and that their union is X. Let B be the next bag and $P(B)$ its parent. We claim that

$$R_B \cap X = \varnothing.$$

Suppose instead that some element x lies in both R_B and X. Then $x \in B$ and x belongs to some previously processed bag C. Because the processed bags form a rooted connected subtree not containing descendants of B that have not yet been processed, the unique path from B to C passes through $P(B)$. By running intersection, every bag on that path contains x, and in particular $x \in P(B)$. But $x \in R_B = B \setminus (B \cap P(B))$ implies $x \notin P(B)$, a contradiction.

Therefore R_B is disjoint from the current accumulator, and

$$X_{\text{new}} = X \mathbin{\dot{\cup}} R_B$$

is legal. Iterating over all non-root bags gives

$$X_{\text{final}} = B_r \mathbin{\dot{\cup}} \dot{\bigcup_{B \neq B_r}} \Big(B \mathbin{\dot{\setminus}} (B \cap P(B)) \Big) = \bigcup_{B \in \mathcal{B}} B.$$

Corollary 2 identifies the last union with $\bigcup \mathcal{F}$. Hence $\bigcup \mathcal{F} \in \bullet(\mathcal{N})$.

The expression is left-linear because one keeps a single accumulator and adjoins one residue at a time. □

Corollary 4 (Linear-size certificate)**.** *Under the hypotheses of Theorem 3, a witness has $O(m)$ structural size, where $m = |\mathcal{B}|$.*

Proof. The certificate consists of the m maximal bags, the $m - 1$ tree edges, and the separator attached to each edge. There is one root leaf and, for each non-root bag, at most one subset complement followed by one disjoint union. □

Theorem 3 is most naturally understood as an acyclic-hypergraph statement. Regard the maximal non-cancelling sets $\mathcal{B}$ as hyperedges on the common ground set. A join tree is precisely a tree representation in which the hyperedges containing any fixed vertex remain connected. This is the same running-intersection geometry that underlies classical acyclic database schemes and join-tree algorithms [3].

The separator $D_B = B \cap P(B)$ has two roles. Combinatorially, it is the complete interface through which the child bag B can overlap the already processed side of the tree. Algebraically, the separator-noncancelling assumption ensures that this interface is available as a legal operand in the NCI algebra. Subtracting it leaves the residue

$$R_B = B \setminus D_B,$$

which contains exactly the elements first introduced on the child side.

The running-intersection property is stronger than merely requiring adjacent overlaps to be controlled. It implies the following locality statement.

Proposition 5 (Residue locality)**.** *Let T be a join tree rooted at B_r. For every non-root bag B,*

$$B \setminus P(B)$$

is disjoint from every bag outside the rooted subtree of B.

Proof. If $x \in B \setminus P(B)$ also belonged to a bag C outside the rooted subtree of B, then the path from B to C would pass through $P(B)$. Running intersection would force $x \in P(B)$, a contradiction. □

Thus the proof of Theorem 3 can be read as a leaf-elimination argument on an acyclic hypergraph. Each elimination step removes a child hyperedge after separating the part already represented by its parent interface. No global rearrangement is required.

This also clarifies the likely source of difficulty beyond the theorem. In a cyclic overlap pattern, a bag can interact with several previously assembled regions through interfaces that are not nested in a single parent separator. Even when each pairwise overlap exists as an intersection, cancellations may prevent the required interfaces from being available as legal leaves. The theorem therefore isolates a clean mechanism by which acyclicity and non-cancellation cooperate.

5 Relation to the NCI Literature

Amarilli, Monet, and Suciu formulate NCI both in set-theoretic language and in terms of principal downsets of Boolean lattices [1]. They prove the conjecture in two notable regimes: when there are no cancelling intersections, and when there is exactly one cancelling intersection. They also investigate stronger syntactic restrictions on the witnessing expression and report exhaustive computations for small ground sets.

Theorem 3 is orthogonal to those cancellation-count regimes. It allows multiple cancelling intersections; what matters is that the *maximal surviving intersections* form an acyclic hypergraph and that the separators appearing in one join tree survive cancellation. In particular, the theorem can apply when many intersections cancel, provided none of the separators needed by the chosen join tree does.

The result also interacts sharply with the left-linear question. Amarilli–Monet–Suciu proposed left-linearity as a possible strengthening, but Wilhelm recently proved that unrestricted left-linear witnesses do not always exist [2]. Theorem 3 does not contradict that negative result: it gives a structural subclass on which left-linearity is guaranteed. Hence the failure of left-linearity in general must occur outside at least one of the two ingredients used here: join-tree acyclicity or non-cancelling availability of the relevant separators.

A useful way to summarize the comparison is

no cancellations or one cancellation (known cases from [1])	versus	arbitrarily many cancellations allowed, subject to acyclic maximal structure and surviving separators.

Neither condition contains the other in an obvious way; they control different aspects of the instance.

Once $\mathcal{N}$ and its maximal members are given, a certificate for Theorem 3 is particularly simple. One supplies a tree T on $\mathcal{B}$ and verifies:

1. T has $m-1$ edges and is connected;
2. for every ground element x, the bags containing x induce a connected subtree;
3. for every tree edge BB', the separator $B \cap B'$ is empty or belongs to $\mathcal{N}$;
4. the rooted residue construction evaluates to $\bigcup \mathcal{F}$.

All four checks are polynomial in the explicit input size. In particular, verification does not require reproducing whatever search procedure found the tree. This proof-certificate separation is useful computationally: a sophisticated private search engine may be used to discover promising instances or join trees, while the published mathematical artifact can remain a small transparent verifier.

One can also search for a valid join tree directly. Build the graph whose vertices are the maximal bags and whose candidate edge BB' is allowed only when $B \cap B'$ is empty or belongs to $\mathcal{N}$. A valid certificate is a spanning tree of this graph that satisfies running intersection. For the modest bag counts used in the supporting experiments, direct enumeration is sufficient; larger instances invite standard join-tree recognition algorithms from acyclic-hypergraph theory [3].

7 Supporting Computational Verification

The theorem above does not rely on computation. We nevertheless implemented a public stress-test harness to check the construction over a large generated corpus. The program:

1. generates pairwise incomparable finite families with a planted join-tree geometry;
2. computes all collected inclusion–exclusion coefficients;
3. extracts $\mathcal{N}$ and its maximal members;
4. searches for a separator-noncancelling join tree;
5. constructs the left-linear witness from Theorem 3; and
6. evaluates the expression independently.

The experiment used a logical candidate-space parameter of 10^6 and actually sampled 50,000 independently generated families. This distinction is important: the theorem verification count is 50,000, not one million exhaustive cases.

Quantity	Reference run	Carnets/SciPy replication
Logical candidate-space parameter	1,000,000	1,000,000
Families actually sampled	50,000	50,000
Theorem-eligible families	50,000	50,000
Verified certificates	50,000	50,000
Maximum $\lvert\mathcal{N}\rvert$	11	11
Maximum collected zero intersections	11	11
Elapsed seconds	12.393176	8.862045
All eligible pass	True	True

Table 1: Supporting computational verification.

Both executions selected the same maximal-cancellation witness signature:

```
3cacc9c316bd113833052e2e4556298eb8a9a45bf4e5f14b7d0051a37af43750
```

The computation should be interpreted as a falsification attempt and implementation check, not as evidence replacing the proof.

8 Reference Verification Code

The following public code is intentionally self-contained and exposes no private search or acceleration implementation. If a compatible external hash module is present it may be used for repeated canonical signatures; otherwise SHA-256

```
#!/usr/bin/env python3
from __future__ import print_function
import argparse, hashlib, itertools, random, time

try:
    import hkd_hash as _hash_mod
except Exception:
    _hash_mod = None

def stable_hash(obj):
    if _hash_mod is not None:
        fn = getattr(_hash_mod, "hash", None)
        if callable(fn):
            try:
                return fn(obj)
            except Exception:
                pass
    return hashlib.sha256(repr(obj).encode("utf-8")).hexdigest()

def collected_mu(F):
    n = len(F)
    mu = {}
    for mask in range(1, 1 << n):
        idx = [i for i in range(n) if (mask >> i) & 1]
        z = set(F[idx[0]])
        for i in idx[1:]:
            z.intersection_update(F[i])
        z = frozenset(z)
        mu[z] = mu.get(z, 0) + (1 if len(idx) & 1 else -1)
    return mu

def noncancelling(F):
    mu = collected_mu(F)
    return set(x for x, c in mu.items() if c != 0), mu

def maximal_sets(N):
    return [A for A in N if not any(A < B for B in N)]

def running_intersection(bags, edges):
    adj = [[] for _ in bags]
    for a, b in edges:
        adj[a].append(b); adj[b].append(a)
    universe = set().union(*bags) if bags else set()
    for x in universe:
        nodes = [i for i, B in enumerate(bags) if x in B]
        if len(nodes) <= 1:
            continue
        allowed, seen = set(nodes), {nodes[0]}
        stack = [nodes[0]]
        while stack:
            u = stack.pop()
            for v in adj[u]:
                if v in allowed and v not in seen:
                    seen.add(v); stack.append(v)
        if seen != allowed:
            return False
    return True

def find_join_tree(bags, N):
    m = len(bags)
    if m <= 1:
        return []
    allowed = []
    for i, j in itertools.combinations(range(m), 2):
        sep = bags[i] & bags[j]
        if not sep or frozenset(sep) in N:
            allowed.append((i, j))
    for comb in itertools.combinations(allowed, m - 1):
        parent = list(range(m))
        def find(x):
            while parent[x] != x:
                parent[x] = parent[parent[x]]
                x = parent[x]
            return x
        ok = True
        for a, b in comb:
            ra, rb = find(a), find(b)
            if ra == rb:
                ok = False; break
            parent[ra] = rb
        if ok and running_intersection(bags, comb):
            return list(comb)
    return None

def construct_expression(bags, edges, N):
```

```
        return frozenset(), ("EMPTY",), []
    adj = [[] for _ in bags]
    for a, b in edges:
        adj[a].append(b); adj[b].append(a)
    root = 0
    par = {root: -1}; order = [root]
    for u in order:
        for v in adj[u]:
            if v not in par:
                par[v] = u; order.append(v)
    X = set(bags[root])
    expr = ("LEAF", frozenset(bags[root]))
    steps = []
    for v in order[1:]:
        p = par[v]
        sep = frozenset(bags[v] & bags[p])
        if sep and sep not in N:
            raise AssertionError("separator not non-cancelling")
        residue = frozenset(bags[v] - sep)
        residue_expr = (("DIFF", ("LEAF", frozenset(bags[v])),
                         ("LEAF", sep)) if sep
                        else ("LEAF", frozenset(bags[v])))
        if X & set(residue):
            raise AssertionError("running-intersection disjointness failed")
        X.update(residue)
        expr = ("DUNION", expr, residue_expr)
        steps.append((v, p, sep, residue))
    return frozenset(X), expr, steps

def eval_expr(e, N):
    typ = e[0]
    if typ == "LEAF":
        s = e[1]
        if s and s not in N:
            raise AssertionError("leaf not in N")
        return s
    if typ == "DIFF":
        A, B = eval_expr(e[1], N), eval_expr(e[2], N)
        if not B.issubset(A):
            raise AssertionError("illegal subset complement")
        return A - B
    if typ == "DUNION":
        A, B = eval_expr(e[1], N), eval_expr(e[2], N)
        if A & B:
            raise AssertionError("illegal disjoint union")
        return A | B
    if typ == "EMPTY":
        return frozenset()
    raise AssertionError("unknown expression")

def random_join_tree_family(rng, bags_n=5, core_n=3, private_n=2):
    parents = [-1]
    for i in range(1, bags_n):
        parents.append(rng.randrange(i))
    B = [set() for _ in range(bags_n)]
    for g in range(core_n):
        token = ("g", g, rng.randrange(10**9))
        chosen = [i for i in range(bags_n) if rng.random() < 0.45]
        if not chosen:
            chosen = [rng.randrange(bags_n)]
        closure = set()
        for v in chosen:
            while v != -1:
                closure.add(v); v = parents[v]
        for v in closure:
            B[v].add(token)
    for i in range(bags_n):
        for k in range(private_n):
            B[i].add(("p", i, k, rng.randrange(10**9)))
    for i in range(1, bags_n):
        p = parents[i]
        for k in range(1 + rng.randrange(2)):
            t = ("s", i, k, rng.randrange(10**9))
            B[i].add(t); B[p].add(t)
    return [frozenset(x) for x in B]

def test_family(F):
    for i in range(len(F)):
        for j in range(i):
            if F[i] <= F[j] or F[j] <= F[i]:
                return {"eligible": False}
    N, mu = noncancelling(F)
    bags = maximal_sets(N)
    T = find_join_tree(bags, N)
    if T is None:
        return {"eligible": False}
```

```
    got, expr, steps = construct_expression(bags, T, N)
    verified = (eval_expr(expr, N) == target == got)
    sig_data = tuple(sorted(tuple(sorted(map(repr, x))) for x in N))
    return {"eligible": True, "verified": verified,
            "bags": len(bags), "nci": len(N),
            "zeros": sum(1 for c in mu.values() if c == 0),
            "tree_edges": len(T), "steps": steps,
            "signature": stable_hash(sig_data)}

def main():
    ap = argparse.ArgumentParser()
    ap.add_argument("--samples", type=int, default=50000)
    ap.add_argument("--logical-samples", type=int, default=1000000)
    ap.add_argument("--seed", type=int, default=20260823)
    args, unknown = ap.parse_known_args()
    if unknown:
        print("ignored_args=%r" % (unknown,))
    rng = random.Random(args.seed)
    tested = eligible = passed = 0
    max_nci = max_zeros = 0
    witness = None
    t0 = time.perf_counter()
    for q in range(args.samples):
        F = random_join_tree_family(rng,
                bags_n=3 + rng.randrange(4),
                core_n=1 + rng.randrange(4),
                private_n=1 + rng.randrange(3))
        r = test_family(F); tested += 1
        if r.get("eligible"):
            eligible += 1
            max_nci = max(max_nci, r["nci"])
            max_zeros = max(max_zeros, r["zeros"])
            if r["verified"]:
                passed += 1
                if witness is None or r["zeros"] > witness[1]["zeros"]:
                    witness = (F, r)
            else:
                raise AssertionError("certificate failure")
    dt = time.perf_counter() - t0
    print("HKD_NCI_JOIN_TREE_RESEARCH")
    print("target_conjecture=NON_CANCELLING_INTERSECTIONS")
    print("theorem=JOIN_TREE_SEPARATOR_NONCANCELLING_CLASS")
    print("logical_candidate_space=%d" % args.logical_samples)
    print("sampled_families=%d" % tested)
    print("theorem_eligible=%d" % eligible)
    print("verified_certificates=%d" % passed)
    print("max_noncancelling_intersections=%d" % max_nci)
    print("max_collected_zero_intersections=%d" % max_zeros)
    print("elapsed_seconds=%.6f" % dt)
    print("all_eligible_pass=%s" % (eligible == passed))
    if witness:
        F, r = witness
        print("witness_bags=%d" % r["bags"])
        print("witness_nci=%d" % r["nci"])
        print("witness_zeros=%d" % r["zeros"])
        print("witness_tree_edges=%d" % r["tree_edges"])
        print("witness_signature=%s" % r["signature"])
    print("FULL_NCI_CONJECTURE_SOLVED=False")
    print("PASS=%s" % (eligible > 0 and eligible == passed))

if __name__ == "__main__":
    main()
```

9 Replicated Program Output

The following is the independent Carnets/SciPy replication. The Jupyter kernel arguments are accepted using `parse_known_args()`.

```
ignored_args=['-f',
'/private/var/mobile/Containers/Data/Application/
9A4EF0F8-2599-4B31-9601-82A90BD5E0CE/Library/jupyter/runtime/
kernel-83933bee-6ac4-409a-88d2-5a6fdd333530.json']

HKD_NCI_JOIN_TREE_RESEARCH
target_conjecture=NON_CANCELLING_INTERSECTIONS
theorem=JOIN_TREE_SEPARATOR_NONCANCELLING_CLASS
```

```
sampled_families=50000
theorem_eligible=50000
verified_certificates=50000
max_noncancelling_intersections=11
max_collected_zero_intersections=11
elapsed_seconds=8.862045
all_eligible_pass=True
witness_bags=6
witness_nci=11
witness_zeros=11
witness_tree_edges=5
witness_signature=
3cacc9c316bd113833052e2e4556298eb8a9a45bf4e5f14b7d0051a37af43750
FULL_NCI_CONJECTURE_SOLVED=False
PASS=True
```

10 Further Questions

The theorem suggests several concrete extensions.

First, the separator requirement could be weakened. It may suffice that a cancelling separator is itself constructible from smaller non-cancelling intersections. A recursive separator theorem of this form would extend the class while preserving the same tree geometry.

Second, one can ask whether some bounded form of cyclicity can be handled by introducing a small number of auxiliary separators. This would interpolate between the acyclic case proved here and the unrestricted NCI problem.

Third, the interaction with Wilhelm's counterexample raises a structural question: can every obstruction to left-linearity be localized to a cyclic core or to a separator whose Möbius coefficient vanishes? Theorem 3 shows that neither phenomenon can occur in a separator-noncancelling join-tree instance.

Finally, from an algorithmic viewpoint, the certificate separation suggests searching directly over canonical intersection structures rather than over raw set families. Such a search can be heavily optimized privately, while any resulting theorem-eligible certificate remains independently checkable by the simple public verifier above.

11 Limitations and Claim Boundary

This paper does not claim to solve the full NCI Conjecture. The proved implication is

$$\text{separator-noncancelling join tree} \quad \Longrightarrow \quad \text{NCI construction.}$$

The left-linear conclusion is likewise restricted to this class and is consistent with the general negative result of Wilhelm [2].

The 50,000-family experiment is not exhaustive over all families of any fixed ground-set size. It verifies 50,000 generated instances satisfying the theorem's structural hypotheses. The parameter `logical_candidate_space=1000000` is an experimental configuration label and should not be interpreted as one million exhaustively checked families.

12 Conclusion

We proved a constructive acyclic-hypergraph sufficient condition for the Non-Cancelling Intersections Conjecture. If the maximal non-cancelling intersections admit a join tree whose nonempty edge separators are themselves non-cancelling, then the target union has the explicit left-linear representation

$$\boxed{\bigcup \mathcal{F} = B_r \,\dot\cup \dot{\bigcup_{B \neq B_r}} \Big(B \,\dot\setminus\, (B \cap P(B)) \Big).}$$

The proof is a forced residue decomposition: running intersection limits each child residue disjoint from the already assembled side, while separator non-cancellation makes each residue legally constructible.

The theorem yields a linear-size structural certificate and a polynomial-time verifier once a join tree is supplied. It also identifies a natural positive left-linear class despite the failure of unrestricted left-linearity. Supporting computation verified the construction on 50,000 independently generated theorem-eligible families in both a reference run and an independent Carnets/SciPy replication.

References

[1] A. Amarilli, M. Monet, and D. Suciu, "The Non-Cancelling Intersections Conjecture," *arXiv preprint arXiv:2401.16210*, 2024. Available: `https://arxiv.org/abs/2401.16210`.

[2] H. Wilhelm, "The Non-Cancelling-Intersections Conjecture Fails for Left-Linear Trees," *arXiv preprint arXiv:2608.19414*, 2026. Available: `https://arxiv.org/abs/2608.19414`.

[3] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis, "On the Desirability of Acyclic Database Schemes," *Journal of the ACM*, vol. 30, no. 3, pp. 479–513, 1983.

[4] G.-C. Rota, "On the Foundations of Combinatorial Theory I. Theory of Möbius Functions," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 2, pp. 340–368, 1964.

[5] R. P. Stanley, *Enumerative Combinatorics, Volume 1*, 2nd ed. Cambridge University Press, 2011.