Deciding the Closure of Inconsistent Rooted Triples Is NP-Complete

Matthew P. Johnson
Department of Computer Science, Lehman College
Ph.D. Program in Computer Science, The Graduate Center
City University of New York, USA

Abstract

Interpreting three-leaf binary trees or *rooted triples* as constraints yields an entailment relation, whereby binary trees satisfying some rooted triples must also thus satisfy others, and thence a closure operator, which is known to be polynomial-time computable. This is extended to inconsistent triple sets by defining that a triple is entailed by such a set if it is entailed by any consistent subset of it.

Determining whether the closure of an inconsistent rooted triple set can be computed in polynomial time was posed as an open problem in the Isaac Newton Institute's "Phylogenetics" program in 2007. It appears (as **NC4**) in a collection of such open problems maintained by Mike Steel, and it is the last of that collection's five problems concerning computational complexity to have remained open. We resolve the complexity of computing this closure, proving that its decision version is NP-Complete.

In the process, we also prove that detecting the existence of *any* acyclic B-hyperpath (from specified source to destination) is NP-Complete, in a significantly narrower special case than the version whose *minimization* problem was recently proven NP-hard by Ritz et al. This implies it is NP-hard to approximate (our special case of) their minimization problem to within *any* factor.

2012 ACM Subject Classification Mathematics of computing → Trees, Mathematics of computing → Hypergraphs, Theory of computation → Problems, reductions and completeness, Applied computing → Molecular evolution

Keywords and phrases phylogenetic trees, rooted triple entailment, NP-Completeness, directed hypergraphs, acyclic induced subgraphs, computational complexity

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.12

Acknowledgements This work was supported in part by NSF award INSPIRE-1547205, and by the Sloan Foundation via a CUNY Junior Faculty Research Award.

1 Introduction

We investigate the computational complexity of a problem in which, based on a given collection of relationships holding between the leaves of a hypothetical (rooted) binary tree T, the task is to infer whatever additional relationships (of the same form) must also hold between T's leaves as a consequence. Various problems in phylogenetic tree reconstruction involve inference of this kind. The specific relationship form in question here, obtaining between some three leaves p, q, o and denoted $pq|o$, is that of the *path between p and q* being node-disjoint from the *path between o and the root*, or equivalently, of the *lowest common ancestor (lca) of p and q* not being an ancestor of o. This relationship is modeled as a *rooted triple*, i.e., the (rooted, full) binary tree on leaves p, q, o in which p and q are siblings, and their parent and o are both children of the root. Then $pq|o$ holding in T is equivalent to having the *subtree of T induced by p, q, o* be homeomorphic to $pq|o$'s corresponding three-leaf binary tree.

The problem of computing the set of all rooted triples entailed by a given triple set R' (its *closure* $\overline{R'}$) is known to be polynomial-time computable by, e.g., Aho et al.'s BUILD algorithm [6, 1] if R' is *consistent*, i.e., if there exists a binary tree satisfying all triples in R'.

If a rooted triple set R is inconsistent, then a given triple is said to be entailed by R if it is entailed by *any* consistent subset $R' \subset R$. That is, the closure $\overline{R}$ equals the union of the closures of all R's *consistent* subsets. Thus the naive brute-force algorithm for computing $\overline{R}$ suggested by the definition is exponential-time in $|R|$.

Determining the complexity of the problem of computing $\overline{R}$ was posed in the Isaac Newton Institute's "Phylogenetics" program in 2007 [9], and it appears (as **NC4**) in a collection of such open problems maintained by Mike Steel [13]. That collection's other four problems concerning computational complexity were all solved by 2009 or 2010, but **NC4** has remained open. We resolve the complexity of computing $\overline{R}$, proving that it is NP-hard. In particular, we prove that its decision version, i.e., deciding whether a given rooted triple is entailed by R, is NP-Complete.

In the process, we also obtain stronger hardness results for a problem concerning *acyclic B-hyperpaths*, a directed hypergraph problem that has recently been applied to another computational biology application, but interestingly one unrelated to phylogenetic trees and rooted triples: *signaling pathways*, the sequences of chemical reactions through which cells respond to signals from their environment (see Ritz et al. [11]).

Specifically, we prove that detecting the existence of *any* acyclic B-hyperpath (between specified source and destination) is NP-Complete, in a significantly narrower special case (viz., the case in which every hyperarc has one tail and two heads) than the version whose *minimization* problem was recently proven NP-hard by Ritz et al. This immediately implies it is NP-hard to approximate (our special case of) their minimization problem to within *any* factor. Moreover, even if we restrict ourselves to feasible problem instances (i.e., those for which there exists at least one such acyclic B-hyperpath), we show that this "promise problem" [8] special case is NP-hard to approximate to within factor $|V|^{1-\epsilon}$ for all $\epsilon > 0$.

Related Work

Inference of new triples from a given set of rooted triples holding in a binary tree was studied by Bryant and Steel [6, 5], who proved many results on problems involving rooted triples, as well as quartets, and defined the closure of an inconsistent triple set. The polynomial-time BUILD algorithm of Aho et al. [1] (as well as subsequent extensions and speedups) can be used to construct a tree satisfying all triples in R (and to obtain the closure $\overline{R}$), or else to conclude than none exists.

Gallo et al. [7] defined a number of basic concepts involving paths and cycles in directed hypergraphs, including B-connectivity. Ausiello et al. [2] studied path and cycle problems algorithmically in directed hypergraphs and showed, via a simple reduction from Set Cover, that deciding whether there exists a B-hyperpath from specified source to destination with $\leq \ell$ hyperarcs is NP-Complete. Ritz et al. [11] recently studied a problem involving "signaling hypergraphs", which are directed hypergraphs that can contain "hypernodes". They modify Ausiello et al.'s hardness reduction from Set Cover to show that deciding the existence of a length$\leq \ell$ B-hyperpath is NP-Complete already in the special case of directed hypergraphs each of whose hyperarcs has at most 3 head nodes and at most 3 tail nodes (due to Set Cover becoming hard once sets of size 3 are permitted). Ritz et al.'s hardness proof actually does not use the fact that their problem formulation requires the computed B-hyperpath to be acyclic. Because the entire directed hypergraph they construct is (like Ausiello et al.'s) always acyclic, their proof provides hardness regardless of whether the formulation includes

an acyclicity constraint. This constraint is essential to our hardness proof, however, so our result does not rule out the possibility that a B-hyperpath minimization problem formulation *without an acyclicity requirement* would be easier to approximate.

2 Preliminaries

2.1 Rooted Triples

▶ **Definition 1.** For any nodes u, v of a rooted binary tree (or simply a *tree*):

- $v \leq u$ denotes that v is a *descendent* of u (and u is an *ancestor* of v), i.e., u appears on the path from v to the root; $v < u$ denotes that v is a *proper descendent* of u (and u is a *proper ancestor* of v), i.e., $v \leq u$ and $v \neq u$.
- uv denotes their *lowest common ancestor (lca)*, i.e., the node w of maximum distance from the root that satisfies $w \geq u$ and $w \geq v$.

▶ **Definition 2.**

- A *rooted triple* (or simply a *triple*) $t = (\{p, q\}, o) \in \binom{L}{2} \times L$ (with p, q, o all distinct, for an underlying finite leaf set L) is denoted by the shorthand notation $pq|o$ and represents the constraint: *the path from p to q is node-disjoint from the path from o to the root.*
- The *left-hand side (LHS)* of a triple $pq|o$ is pq, and its *right-hand side (RHS)* is o.
- $L(T)$ denotes the set of leaves of a tree T, and $L(R')$ denotes the set of leaves appearing in any of the triples within a set R', i.e., $L(R') = \bigcup_{pq|o \in R'} \{p, q, o\}$.
- A tree T with $p, q, o \in L(T)$ *displays* the triple $pq|o$ (or, $pq|o$ *holds* in T) if the corresponding constraint holds in T. The set of all triples displayed by T is denoted by $r(T)$. The set of all trees that display *all* triples in R' is denoted by $\langle R' \rangle$. A set of triples R' is *consistent* if $\langle R' \rangle$ is nonempty.

▶ **Definition 3.**

- For a consistent triple set R', a given triple t (which may or may not be a member of R') is *entailed* by R', denoted $R' \vdash t$, if every tree displaying all the triples in R' also displays t, i.e., if t is displayed by every tree in $\langle R' \rangle$. The *closure* $\overline{R'}$ is the set of all triples entailed by R', i.e., $\overline{R'} = \{t : R' \vdash t\}$, which can also be defined as $\overline{R'} = \bigcap_{T \in \langle R' \rangle} r(T)$ [6].
- For an inconsistent triple set R, a given triple t (which may or may not be a member of R) is *entailed* by R, again denoted $R \vdash t$, if there exists a consistent subset $R' \subset R$ that entails t. The closure $\overline{R}$ is again the set of all triples entailed by R, or equivalently the union, taken over every consistent subset $R' \subset R$, of $\overline{R'}$, i.e., $\bigcup_{\text{cons. } R' \subset R} \overline{R'}$.

We first state a few immediate consequences of these definitions.

▶ **Observation 4.**

- *It can happen that $pp' = qq'$ even if $\{p, p'\} \cap \{q, q'\} = \varnothing$.*
- *In any given tree T having $p, q, o \in L(T)$, exactly one of $pq|o$, $po|q$, and $qo|p$ holds.*
- *$pq|o$ iff $qp|o$ iff (path: p to q) $\cap$ (path: o to the root) $= \varnothing$ iff $pq < po = qo$.*
- *Equivalently, the* 3-point condition *for* ultrametrics *[12] holds: for all $p, q, o \in L(T)$, we have $pq < po = qo$ or $oq < op = qp$ or $op < oq = pq$.*
- *Regardless of whether triple set R is consistent, its closure $\overline{R}$ satisfies $R \subseteq \overline{R} \subseteq \binom{L}{2} \times L$, and so $|\overline{R}| = O(|L|^3)$.*

We state the problem formally.

Table 1 Variable name conventions, many of which (also) represent leaves in the triple set R constructed in the reduction. Note that the notation pq (for leaves p, q) is used to denote both $\text{lca}(p, q)$ *and* the hypergraph node whose outgoing hyperarcs represent triples of the form $pq|o$, i.e., those constraining $\text{lca}(p, q)$ from above.

p, q, p', q', o, o'	generic leaf variables, especially in triples' LHSs or RHSs (resp.)	(leaves)
b_i, b'_i, c_j, d_j, etc.	particular leaf names	(leaves)
pq, etc.	lowest common ancestor $\text{lca}(p, q)$ of leaves p, q	(leaf 2-sets)
α, β, γ	leaves of target triple $\alpha\beta\|\gamma$	(leaves)
t	rooted triple, especially of form $p_k q_k \| o_k = u_k \| o_k$	
R or R'	set of triples, especially inconsistent or consistent (resp.)	
L or $L(R)$	set of leaves or set of leaves appearing in members of R (resp.)	(leaf sets)
u, u_k, v, v', v_k, v'_k	hypergraph nodes, especially tail node or head nodes (resp.)	(leaf 2-sets)
pq, etc.	hypergraph node corresponding to leaves p, q	(leaf 2-sets)
$\alpha\beta,\ c_{m+1}\gamma$	source and destination nodes (resp.)	(leaf 2-sets)
a_k	1-2-hyperarc, especially of form $u_k \to \{v_k, v'_k\} = p_k q_k \to \{p_k o_k, q_k o_k\}$, with $k \in [\ell] = \{1, ..., \ell\}$ indicating a_k's position in a path P of length $\|P\| = \ell$	
x_i	ith SAT variable, with $i \in [n]$	
C_j	jth SAT clause, with $j \in [m]$	
$x_i, \bar{x}_i$ or $\tilde{x}_i$	literals (positive, negative or either, resp.) of x_i	
$x_i^j, \bar{x}_i^j$ or $\tilde{x}_i^j$	the appearance (positive, negative or either, resp.) of x_i in C_j	(leaves)
$x_{\hat{w}}^j, \bar{x}_{\hat{w}}^j$ or $\tilde{x}_{\hat{w}}^j$	the wth variable appearance in C_j	(leaves)
$x_{\cdot}^j, \bar{x}_{\cdot}^j$ or $\tilde{x}_{\cdot}^j$	*some* (unspecified) variable appearance in C_j	(leaves)
$y_i^j, \bar{y}_i^j$	helper leaves in x_i gadget for x_i^j and $\bar{x}_i^j$ (resp.)	(leaves)
$\tilde{z}_i^j$	jth element in sequence $b_j, b'_j, \tilde{x}_i^1, \tilde{y}_i^1 ..., \tilde{x}_i^m, \tilde{y}_i^m, b_{j+1}, b'_{j+1}$	(leaves)
F	SAT formula	

Inconsistent Rooted Triple Set Closure

INSTANCE: An inconsistent rooted triple set R.
SOLUTION: R's closure $\overline{R} = \{t : R \vdash t\}$.

By the observation above, computing the closure is equivalent to solving the following decision problem for each of the $O(|L|^3)$ triples $t \in \binom{L}{2} \times L$.

Inconsistent Rooted Triple Set Entailment

INSTANCE: An inconsistent rooted triple set R and a rooted triple t.
QUESTION: Does $R \vdash t$, i.e., does there exists a consistent triple set $R' \subset R$ satisfying $R' \vdash t$?

Although there is no finite set of inference rules that are complete [6], there are only three possible inference rules inferring from two triples [6].

▶ **Definition 5.** The three *dyadic inference rules* ($\forall\, p, q, o, p', o' \in L$) are:

$$\begin{aligned} \{pq|o,\ qp'|o\} &\vdash pp'|o \\ \{pq|o,\ qo|o'\} &\vdash \{pq|o',\ po|o'\} \\ \{pp'|o,\ oo'|p\} &\vdash \{pp'|o',\ oo'|p'\} \end{aligned} \tag{1}$$

A type of graph (distinct from hypergraphs discussed below) that will be used in the hardness proof is the *Ahograph* [1], which is defined for a given triple set R and leaf set L.[1]

▶ **Definition 6.** For a given triple set R and leaf set L, the *Ahograph* $[R, L]$ is the following undirected *edge-labeled* graph:

- its vertex set equals L;
- for every triple $pq|o \in R$, if $p, q, o \in L$, then there exists an $\{p, q\}$ with label o.

For a hypergraph (V, A), the *corresponding Ahograph* is the Ahograph $[\text{triples}(A), V]$.

To avoid confusion with the nodes of the hypergraph, we refer to the Ahograph's nodes and edges as *A-nodes* and *A-edges*.

2.2 Directed Hypergraphs

Definitions of paths and cycles in hypergraphs are subtler and more complicated than the corresponding definitions for graphs (see [10]). We adopt versions of Gallo et al. [7]'s definitions, simplified for the special case in which every hyperarc has exactly one tail and two heads.

▶ **Definition 7.** A *1-2-directed hypergraph* (or simply *hypergraph*) $H = (V, A)$ consists of a set of nodes V and a set of 1-2-hyperarcs A. A 1-2-hyperarc (or 1-2-directed hyperedge[2], or simply *hyperarc* or *arc*) is an ordered pair $a = (u, \{v, v'\}) \in V \times \binom{V}{2}$, with u, v, v' all distinct, which we denote by $u \to \{v, v'\}$. Let $\text{t}(a) = u$ be a's *tail* and $\text{h}(a) = \{v, v'\}$ be a's *heads*. A node with out-degree 0 is a *sink*.

▶ **Definition 8.**

- A *simple path* from u_0 to u_ℓ is a sequence of distinct 1-2-hyperarcs $P = (a_1, ..., a_\ell)$, where $u_0 = \text{t}(a_1)$, $u_\ell \in \text{h}(a_\ell)$ and $\text{t}(a_{k+1}) \in \text{h}(a_k)$ for all $k \in [\ell - 1]$. The length $|P| = \ell$ is the number of arcs.
- A *cycle* is a simple path having $\text{h}(a_\ell) \ni \text{t}(a_1)$. An arc $a_k \in P$ having one of its heads be the tail of some earlier arc $a_{k'}$ of P, i.e., where $\exists a_{k'} \in P : \ k' < k$ and $\text{h}(a_k) \ni \text{t}(a_{k'})$, is a *back-arc*. A simple path is *cycle-free* or *acyclic* if it has no back-arcs, and is *cyclic* otherwise. More generally, a set $A' \subseteq A$ is *cyclic* if it is a superset of some cycle, and *acyclic* otherwise.

▶ **Definition 9.** In *general* directed hypergraphs (i.e., with no restrictions on arcs' numbers of heads and tails), a node v is *B-connected*[3] to u_0 if $v = u_0$ or (generating such B-connected nodes bottom-up, through repeated application of this definition) if there is a hyperarc a with $v \in \text{h}(a)$ and every node $\text{t}(a)$ is B-connected to u_0. A path P from u_0 to u_ℓ is a *B-hyperpath* if u_ℓ is B-connected to u_0 (using only the arcs $a \in P$).

Due to the following observation, for the remainder of this paper any use of the term "path" will be understood to mean "B-hyperpath".

▶ **Observation 10.** *If all arcs are 1-2-hyperarcs, then every simple path is also a B-hyperpath.*

Via the hypergraph representation used in our hardness proof for Inconsistent Rooted Triple Set Entailment below, we also obtain hardness results for the following problem formulations as a by-product.

---

[1] We choose to define the Ahograph as a multigraph whose edges each have exactly one label, rather than the more common definition as a graph whose edges each have a *set* of labels.

[2] Called a 2-directed F-hyperarc in [14], extending definitions introduced by Gallo et al. [7].

[3] Note also that Gallo et al. [7] defines *B-hyperarc* simply to mean an arc a having $|h(a)| = 1$.

Acyclic B-Hyperpath Existence in a 1-2-Hypergraph
INSTANCE: A 1-2-directed hypergraph $H = (V, A)$ and nodes $u, v \in V$.
QUESTION: Does there exist an acyclic B-hyperpath in H from from u to v?

We want to define an optimization version of the problem where the objective is to minimize path P's length $|P|$, but since a given problem solution may contain no solutions at all (it may be *infeasible*, specifically if v is not B-connected to u), we obtain the following somewhat awkward definition. Note that defining the cost of an infeasible solution to be infinity is consistent with the convention that $\min \varnothing = \infty$.

Min Acyclic B-Hyperpath in a 1-2-Hypergraph
INSTANCE: A 1-2-directed hypergraph $H = (V, A)$ and nodes $u, v \in V$.
SOLUTION: A B-hyperpath P in H.
MEASURE: P's length $|P|$, (i.e., its number of hyperarcs), if P is a *feasible* solution (i.e., an acyclic B-hyperpath from u to v), and otherwise infinity.

Alternatively, we can formulate a "promise problem" [8] special case of the minimization problem, restricted to instances admitting feasible solutions.

Min Acyclic B-Hyperpath in a B-Connected 1-2-Hypergraph
INSTANCE: A 1-2-directed hypergraph $H = (V, A)$ and nodes $u, v \in V$, where the v is B-connected to u.
SOLUTION: An acyclic B-hyperpath P in H from u to v.
MEASURE: P's length $|P|$.

3 The Construction

3.1 High-level Strategy

We will prove that INCONSISTENT ROOTED TRIPLE SET ENTAILMENT is NP-Complete by reduction from 3SAT, using a construction similar to that of [3] (see also [4]) for the problem of deciding whether a specified pair of nodes in a directed graph are connected by an *induced* path.[4] So, given a SAT formula F, we must construct a problem instance (R, t) such that $R \vdash t$ iff F is satisfiable. Intuitively, we want to define R in such a way that it will be representable as a graph (or rather, as a directed hypergraph), whose behavior will mimic that of the induced subgraph problem.

In slightly more detail, the instance (R, t) that we define based on F will have a structure that makes it representable as a certain directed hypergraph. This hypergraph (see Fig. 1) will play an intermediate role between (R, t) and F, yielding a two-step reduction between the three problems. In particular, we will show:

1. A path P (from $\alpha\beta$ to $c_{m+1}\gamma$) determines a truth assignment $\mathbf{v}(\cdot)$, and vice versa.
2. P will be acyclic iff $\mathbf{v}(\cdot)$ satisfies F.
3. An acyclic path P (or an acyclic superset of it) determines a consistent subset $R' \subset R$ entailing $t = \alpha\beta|\gamma$, and vice versa.
4. Hence R' will be consistent and entail $\alpha\beta|\gamma$ iff P is acyclic iff $\mathbf{v}(\cdot)$ satisfies F.

[4] That problem becomes trivial if either the graph is undirected or the *induced* constraint is removed.

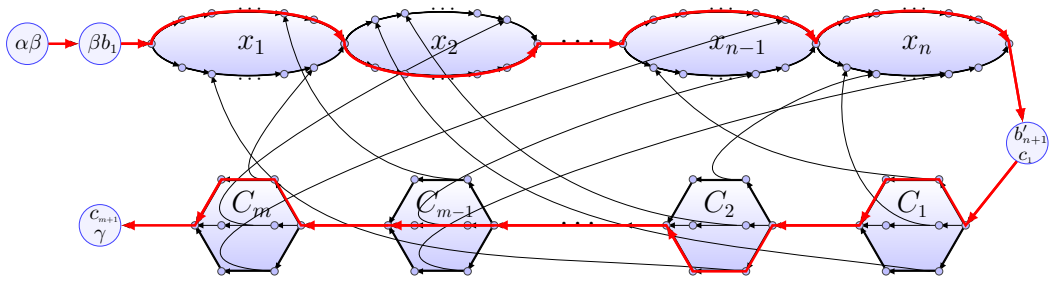


Figure 1 Construction overview, with the path P from $\alpha\beta$ to $c_{m+1}\gamma$ shown in red. Each ellipse represents the gadget for one variable x_i (see Fig. 2a), and each hexagon represents the gadget for one clause C_j (see Fig. 2b). (Sink nodes are omitted for clarity.) The path shown corresponds to a truth assignment in which x_2 is true and x_1, x_3, x_4 are false. For example, the path shown takes x_1's *positive* (upper) side, passing through its *positive* nodes, which *renders x_1's positive appearances unusable*, thus setting x_1 to *false*. C_m's upper *witness path* points to x_1's *negative* (lower) side, indicating that x_1's appearance in C_m is *negative*. Thus x_1 being false satisfies C_m.

The challenge we face is designing a construction that will force cycles to autonomously result from non-satisfiable formulas (mimicking the logic of an *induced* subgraph) is that the definition of entailment of a triple t from an inconsistent set R allows us to pick and choose among the members of R, selecting any consistent subset as the witness to t's entailment, seemingly indicating that any troublesome members of R corresponding to back-arcs causing a cycle could simply be omitted – *independently* of our choices when selecting the triples that we *are* relying on.

The way we disallow this freedom is that we model a rooted triple not as a directed edge in a graph but as a directed *hyperedge*, pointing from one tail node to two head nodes. Although the definition of entailment from an inconsistent triple set R means we can omit any hyperarc we like in defining a possible H', we cannot omit *half* a hyperarc: "turning on" a 1-2-hyperarc $u{\rightarrow}\{v, v'\}$ because we want tail u to point to head v also necessarily causes u to point to v'.

For most of the arcs we define in our construction, these second head nodes will be just spinning wheels: sink nodes having no effect, and omitted for clarity from some figures. The important ones are those in which tail u and one head v both lie in a clause gadget and the other head v' lies in a variable gadget.

3.2 Identifying Rooted Triples and 1-2-Hyperarcs

A core idea of our construction and proof is a correspondence between rooted triples and H's hyperarcs (all 1-2-hyperarcs), which renders them mutually definable in terms of one anther. Each of H's nodes will be identified with an unordered pair of leaves $\{p, q\} \in \binom{L}{2}$ (written for convenience pq), and each of its hyperarcs will have structure of the form $pq{\rightarrow}\{po, qo\}$, with p, q, o all distinct. That is, *each of an arc $u{\rightarrow}\{v, v'\}$'s two heads v, v' will contain one of the tail u's two leaves plus a different leaf common to both v and v'*. This structure ensures that each hyperarc encodes a rooted triple, rather than a constraint of the more general form $pp' < qq'$ Thus we can write $A = \{pq{\rightarrow}\{po, qo\} : pq|o \in R\}$ or $R = \{pq|o : pq{\rightarrow}\{po, qo\} \in A\}$. Indeed, we can simply *identify them with one another* as follows.

▶ **Definition 11.** For a triple $pq|o$, the corresponding hyperarc is $\text{arc}(pq|o) = pq{\rightarrow}\{po, qo\}$; conversely, for a 1-2-hyperarc $pq{\rightarrow}\{po, qo\}$, the corresponding triple is $\text{triple}(pq{\rightarrow}\{po, qo\}) = pq|o$. For a triple set R', we write $\text{arcs}(R')$ to denote the same set R', with but its members *treated as arcs*, and similarly in reverse, for an arc set A', we write $\text{triples}(A')$.

Given this, we can also give a more abstract correspondence.

▶ **Definition 12.** For a 1-2-hyperarc $u{\to}\{v, v'\}$, the corresponding triple is $\text{triple}(u{\to}\{v, v'\}) = v \oplus v' | v \cap v'$, where $\oplus$ denotes symmetric difference. We also combine the two models' syntax, writing $u|o$ to denote $pq|o$ when $u = pq$, i.e., when hyperarc $u{\to}\{v, v'\} = \text{arc}(pq|o)$.

This leads to the following equivalent restatements of the second dyadic inference rule (recall Def. 5) in forms that will sometimes be more convenient.

▶ **Observation 13.** *The first inference of dyadic inference rule (1) can be stated as:*

$$\begin{aligned} \{pq{\to}\{po, qo\},\ qo{\to}\{qo', oo'\}\} \ &\vdash\ pq{\to}\{po', qo'\} \quad (\forall\, p, q, o, o' \in L) \\ \{u_{k-1}|o,\ u_k|o'\} \ &\vdash\ u_{k-1}|o' \qquad\quad (\forall\, u_{k-1}, u_k \in V, o \in u_k \ s.t.\ |u_{k-1} \cap u_k| = 1) \end{aligned} \tag{2}$$

We emphasize again the following two related facts about the meaning of an arc $pq{\to}\{po, qo\} \in A$:

1. If T is a tree with $p, q, o \in L(T)$ and $pq|o \in r(T)$, then lowest common ancestors po and qo are equal, i.e., they refer to the same node in T.
2. Yet po and qo are two distinct A-nodes (in V) of the hypergraph H.

That is, "turning on" triple $pq|o$ (by adding it to the triple set R') has the effect of causing the *hypergraph nodes* po and qo to thence refer to the same *tree node* (in any tree displaying R').

3.3 Defining L and R

Let the SAT formula F on variables $x_1, ..., x_n$ consist of m clauses C_j, each of the form $C_j = (\tilde{x}^j_{i_1} \vee \tilde{x}^j_{i_2} \vee \tilde{x}^3_{i_3})$ or $C_j = (\tilde{x}^j_{i_1} \vee \tilde{x}^j_{i_2})$, where each literal $\tilde{x}^j_i$ has the form either x_i or $\bar{x}_i$ for some i.

We define the leaf set L underlying R as $L = L_1 \cup L_2 \cup L_3 \cup L_4$, where:

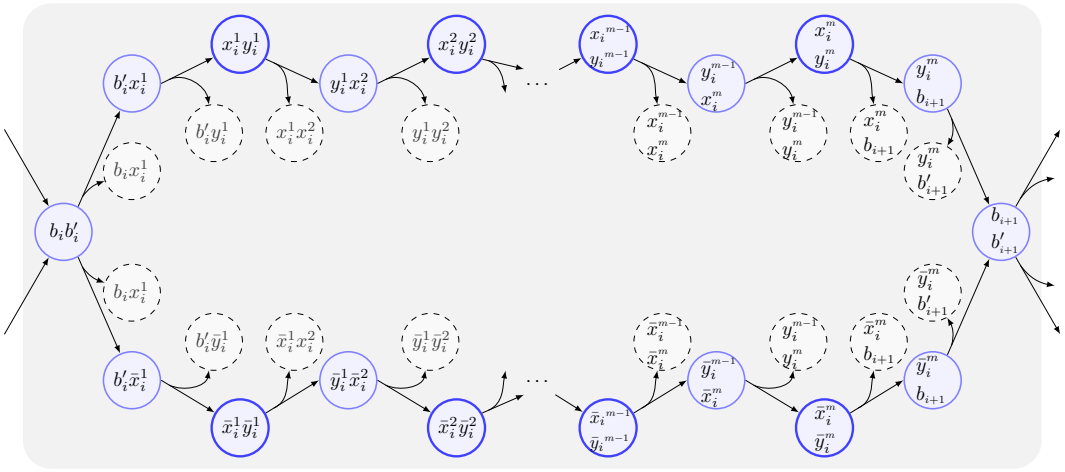
- $L_1 = \bigcup_{i\in[n], j\in[m]} \{x^j_i, \bar{x}^j_i, y^j_i, \bar{y}^j_i\}$ ($4nm$ leaves)[5]
- $L_2 = \bigcup_{i\in[n+1]} \{b_i, b'_i\}$ ($2n + 2$ leaves)
- $L_3 = \bigcup_{j\in[m]} \{c_j, d_j\}$ ($2m$ leaves)
- $L_4 = \{\alpha, \beta, \gamma\}$ (3 leaves)

For each variable x_i in F, we create a gadget consisting of two parallel length-$2m{+}2$ paths intersecting at their first and last nodes but otherwise node-disjoint (see Fig. 2a), where the path taken will determine the variable's truth value. The rooted triples in R corresponding to variable x_i's gadget are:

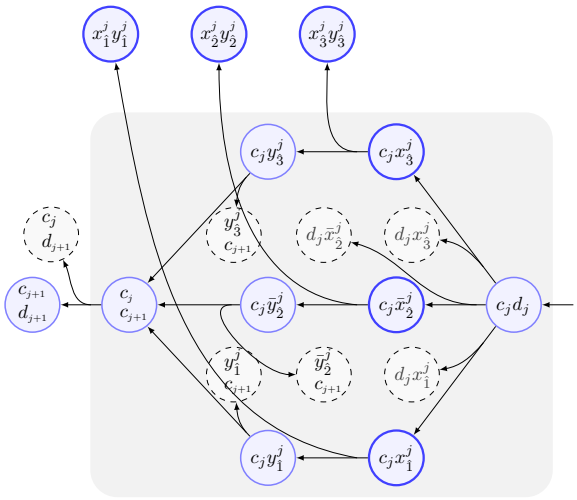
- On its *positive side*:
  $\{b_i b'_i | x^1_i,\ \ b'_i x^1_i | y^1_i,\ \ x^1_i y^1_i | x^2_i,\ \ y^1_i x^2_i | y^2_i,\ \ ...,\ x^{m-1}_i y^{m-1}_i | x^m_i,\ \ y^{m-1}_i x^m_i | y^m_i,\ \ x^m_i y^m_i | b_{i+1},\ \ y^m_i b_{i+1} | b'_{i+1}\}$
- On its *negative side*:
  $\{b_i b'_i | \bar{x}^1_i,\ \ b'_i \bar{x}^1_i | \bar{y}^1_i,\ \ \bar{x}^1_i \bar{y}^1_i | \bar{x}^2_i,\ \ \bar{y}^1_i \bar{x}^2_i | \bar{y}^2_i,\ \ ...,\ \bar{x}^{m-1}_i \bar{y}^{m-1}_i | \bar{x}^m_i,\ \ \bar{y}^{m-1}_i \bar{x}^m_i | \bar{y}^m_i,\ \ \bar{x}^m_i \bar{y}^m_i | b_{i+1},\ \ \bar{x}^m_i b_{i+1} | b'_{i+1}\}$

For each clause $C_j = (\tilde{x}^j_{i_1} \vee \tilde{x}^j_{i_2} \vee \tilde{x}^j_{i_3})$ in F, we create a gadget consisting of three (or two, in the case of a two-literal clause) parallel length-3 paths, intersecting in their first and fourth nodes, followed by one additional (shared) edge (see Fig. 2b), where the path taken (the *witness path*) will correspond to which of C_j's literal satisfies the clause (or one among them, in the case of multiple true literals). The second node of C_j's witness path (of the

[5] Alternatively, we could create such nodes only corresponding to actual appearances of variables in clauses, i.e., $L_1 = \bigcup_{i,j:x_i\in C_j} \{x^j_i, y^j_i\} \cup \bigcup_{i,j:\bar{x}_i\in C_j} \{\bar{x}^j_i, \bar{y}^j_i\}$ ($\leq 3m$ leaves).



(a) Variable gadget for x_i. Any path passing through this gadget (drawn left to right) has two options, taking its *negative* (lower) side, making x_i *true*, or its *positive* (higher) side, making x_i *false*. That is, the truth value corresponding to the path is the one making *the literals in the nodes on the unused side* true. Intuitively, *a path traversing one of the gadget's two sides renders all the literals appearing within that side's nodes unusable*. Note that the rightmost node ($b_{i+1}b'_{i+1}$) is also (for each $i < n$) the leftmost node of x_{i+1}'s gadget.



(b) Clause gadget for $C_j = (x^j_{i_1} \vee \bar{x}^j_{i_2} \vee x^j_{i_3})$, which is followed (drawn outside the shaded region) by node $c_{j+1}d_{j+1}$ (or $c_{m+1}\gamma$, in the case of $j = m$). Any path passing through this gadget (drawn right to left) has three options: going *up*, *straight across*, or *down*, each corresponding to one choice among C_j's three possible *witness paths*. The arrow from the witness path's *witness node*, say, $c_j\tilde{x}^j_i$, to a node $\tilde{x}^j_i\tilde{y}^j_i$ lying within *one of the two sides* of x_i's gadget (and outside the shaded region) represents *the appearance of* x_i *in* C_j; the 1-2-hyperarc that arrow is constituent of forces an acyclic path taking this witness path to have taken the *opposite side* of x_i's gadget.

Figure 2 Gadgets used in the reduction. Each pair of arrows drawn forking from the same tail node represents one 1-2-hyperarc. Sink nodes have dashed borders and are shaded lighter (gray) than non-sink nodes (blue). The clause gadget nodes that point to variable gadget nodes and the variable gadget nodes that can be pointed to by them are both drawn with thick borders.

form $c_j\tilde{x}_i^j$, and corresponding to the appearance of literal $\tilde{x}_i$) is its *witness node*. The rooted triples in R corresponding to clause C_j's gadget are:

- $\{c_jd_j|x_i^j,\ \ c_jx_i^j|y_i^j,\ \ c_jy_i^j|c_{j+1}\}$, for each positive appearance of a variable x_i in C_j
- $\{c_jd_j|\bar{x}_i^j,\ \ c_j\bar{x}_i^j|\bar{y}_i^j,\ \ c_j\bar{y}_i^j|c_{j+1}\}$, for each negative appearance of a variable x_i in C_j
- $c_jc_{j+1}|d_{j+1}$, if $j < m$

Finally, R has the following triples connecting the pieces together, connecting the source node $\alpha\beta$ to a chained-together series of variable gadgets, the last of which is connected (via an intermediate node) to the first of a chained-together series of clause gadgets, the last of which is connected to the destination node $c_{m+1}\gamma$:

- $\{\alpha\beta|b_1,\ \ \beta b_1|b_1'\}$
- $\{b_{n+1}b_{n+1}'|c_1,\ \ b_{n+1}'c_1|d_1\}$
- $c_mc_{m+1}|\gamma$

It is important to remember that all these connections are 1-2-hyperarcs. Sometimes both heads will be nodes within variable and clause gadgets, but in most cases one of the two heads will be a sink node whose only role is to permit the hyperarc to conform to the required structure.

4 The Proof

Clearly Inconsistent Rooted Triple Set Entailment is in NP: if we guess the subset $R' \subset R$, then we can verify both that R' is consistent and that $R' \vdash t$ by executing Aho et al. [1]'s polynomial-time BUILD algorithm on R' [6]. Min Acyclic B-Hyperpath in a 1-2-Hypergraph is as well: guess the path, and check that it is acyclic.

Now we prove hardness, arguing that R contains a consistent subset entailing $\alpha\beta|\gamma$ iff H contains an acyclic path P from $\alpha\beta$ to $c_{m+1}\gamma$ iff F admits a satisfying assignment $\mathbf{v}(\cdot)$, in two steps. Due to lack of space, the proofs are deferred to the full version.

4.1 Acyclic Path ⇔ Satisfying Truth Assignment

First we argue that acyclic paths correspond to satisfying truth assignments.

▶ **Lemma 14.** *There is an an acyclic path P from $\alpha\beta$ to $c_{m+1}\gamma$ iff F admits a satisfying truth assignment $\mathbf{v}(\cdot)$.*

Thus we have proven:

▶ **Theorem 15.** Acyclic B-Hyperpath Existence in a 1-2-Hypergraph *is NP-Complete.*

Since an infeasible solution is defined to have infinite cost, an algorithm with *any* approximation factor would allow us to distinguish between positive and negative problem instances, which immediately implies:

▶ **Corollary 16.** *Approximating* Min Acyclic B-Hyperpath in a 1-2-Hypergraph *to within* any *factor is NP-hard.*

Even if we restrict ourselves to problem instances admitting feasible solutions, this "promise problem" [8] special case is hard to approximate within any reasonable factor.

▶ **Corollary 17.** Min Acyclic B-Hyperpath in a B-Connected 1-2-Hypergraph *is NP-hard to approximate to within factor $|V|^{1-\epsilon}$ for all $\epsilon > 0$.*

Second, to extend the reduction to Inconsistent Rooted Triple Set Entailment, we argue that H is a faithful representation of R in the sense that acyclic paths from $\alpha\beta$ to $c_{m+1}\gamma$ (or acyclic supersets of such paths) correspond to consistent subsets entailing $\alpha\beta|\gamma$, and vice versa.

4.2 Consistent Entailing Subset ⇐ Acyclic Path

We prove this direction via two lemmas, proving that the set of triples corresponding to an acyclic path are consistent and entail $\alpha\beta|\gamma$, respectively.

▶ **Lemma 18.** *If there is an acyclic path $P \subseteq A$ from $\alpha\beta$ to $c_{m+1}\gamma$, then $R' = triples(P)$ is consistent.*

▶ **Lemma 19.** *If there is an acyclic path $P \subseteq A$ from $\alpha\beta$ to $c_{m+1}\gamma$, then $R' = triples(P)$ entails $\alpha\beta|\gamma$.*

Thus we have:

▶ **Corollary 20 (⇐).** *If there is an acyclic path $P \subseteq A$ from $\alpha\beta$ to $c_{m+1}\gamma$, then $R' = triples(P)$ is consistent and entails $\alpha\beta|\gamma$.*

4.3 Consistent Entailing Subset ⇒ Acyclic Path

Now we argue for the reverse direction, proving through a series of lemmas that if there is no acyclic $\alpha\beta$–$c_{m+1}\gamma$ path, then there will be no consistent triple subset entailed $\alpha\beta|\gamma$.

▶ **Lemma 21.** *Let $A' \subseteq A$. Suppose there exists a* cyclic *path $P \subseteq A'$ from $\alpha\beta$ to $c_{m+1}\gamma$. Then $R' = triples(A')$ is inconsistent.*

Most of the remainder of this subsection will be dedicated to showing *constructively* that if A' contains no path from $\alpha\beta$ to $c_{m+1}\gamma$ at all, cyclic or otherwise, then R' does not entail $\alpha\beta|\gamma$. We do so by showing that in the case of such a (consistent) R', there exist trees displaying $R' \cup \{\alpha\beta|\gamma\}$. Therefore assume w.l.o.g. that R' is consistent and maximal in the sense that adding any other triple of R to it would either make R' inconsistent or would introduce an $\alpha\beta$–$c_{m+1}\gamma$ path in $A' = \text{arcs}(R')$.

Observe that the missing arcs $A^\times = A - A'$ can be thought of as the (source side to sink side) cross arcs of a cut separating source $\alpha\beta$ and sink $c_{m+1}\gamma$. In the following argument we will refer to hypergraph $H^\gamma = (V \cup \{\gamma\alpha\}, A' \cup \text{arc}(\gamma\alpha|\beta))$ and its corresponding Ahograph G^γ.

Recalling the construction of H, there are three types of places where the absent cross-arcs $A^\times$ could be located: within a clause gadget, within a variable gadget, or elsewhere, i.e., *forced arcs* (viz., connecting arcs $a_1, ..., a_4$ or arcs with tail of the form $c_j c_{j+1}$ following a clause C_j's gadget). There is one special subcase, which we give a name to.

▶ **Definition 22.** We call $A^\times$ *degenerate* if $A^\times$ lies within a variable x_i's gadget, $|A^\times| = 2$, and exactly one of its members has the form $\text{arc}(b_i b_i'|\tilde{x}_i^1)$. (Its other member must by definition lie within the x_i gadget's opposite side.)

We deal with all cases besides an degenerate $A^\times$ in the following lemma.

▶ **Lemma 23.** *Let R' be consistent. Suppose there is no path $P \subseteq A'$ from $\alpha\beta$ to $c_{m+1}\gamma$, and that $A^\times$ is non-degenerate. Then R' does not entail $\alpha\beta|\gamma$.*

The problematic situation is when exactly *one* of the two arcs is outgoing from $b_i b'_i$. In this case, their absence deletes only one of the Ahograph's two A-edges between the pair $\{b_i, b'_i\}$, which does *not* disconnect the graph, meaning BUILD will fail.

We have been arguing that if a consistent R' entails $\alpha\beta|\gamma$ then arcs(R') must contain an acyclic path from $\alpha\beta$ to $c_{m+1}\gamma$. Now we refine this to a slightly weaker (yet strong enough) implication: if a consistent R' entails $\alpha\beta|\gamma$, then a slightly different consistent R^+ will too, and an acyclic path must exist within arcs(R^+).

This implies:

▶ **Corollary 24** (⇒). *If there is a consistent R' entailing $\alpha\beta|\gamma$ then there exists an acyclic path P.*

Combining the Corollary 20 and 24 with Theorem 15, we conclude:

▶ **Theorem 25.** Inconsistent Rooted Triple Set Entailment *is NP-Complete.*

And because computing the closure reduces to deciding whether $R \vdash t$ for $O(|L|^3)$ triples t, we also have:

▶ **Corollary 26.** Inconsistent Rooted Triple Set Closure *is NP-hard.*

References

1 Alfred V. Aho, Yehoshua Sagiv, Thomas G. Szymanski, and Jeffrey D. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.

2 Giorgio Ausiello, Roberto Giaccio, Giuseppe F Italiano, and Umberto Nanni. Optimal traversal of directed hypergraphs. Technical Report TR-92-073, International Computer Science Institute, Berkeley, CA, September 1992.

3 Jørgen Bang-Jensen, Frédéric Havet, and Nicolas Trotignon. Finding an induced subdivision of a digraph. *Theoretical Computer Science*, 443:10–24, 2012.

4 Dan Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Mathematics*, 90(1):85–92, 1991.

5 David Bryant. *Building Trees, Hunting for Trees, and Comparing Trees: Theory and Methods in Phylogenetic Analysis.* PhD thesis, University of Canterbury, 1997.

6 David Bryant and Mike Steel. Extension operations on sets of leaf-labeled trees. *Advances in Applied Mathematics*, 16(4):425–453, 1995.

7 Giorgio Gallo, Giustino Longo, Stefano Pallottino, and Sang Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993.

8 Oded Goldreich. On promise problems: A survey. In *Theoretical Computer Science: Essays in Memory of Shimon Even*, pages 254–290. Springer, 2006.

9 Daniel Huson, Vincent Moulton, and Mike Steel. Final Report for the 'Phylogenetics' Programme. Technical report, Isaac Newton Institute for Mathematical Sciences, February 2008.

10 Lars Relund Nielsen, Daniele Pretolani, and K Andersen. A remark on the definition of a B-hyperpath. Technical report, Department of Operations Research, University of Aarhus, 2001.

11 Anna Ritz, Brendan Avent, and T Murali. Pathway analysis with signaling hypergraphs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(5):1042–1055, September 2017.

12 Charles Semple and Mike Steel. *Phylogenetics*, volume 24 of *Oxford Lecture Series in Mathematics and Its Applications.* Oxford University Press, 2003.

13 Mike Steel. Phylogenetics: Challenges and Conjectures, updated in July 2018. URL: `http://www.math.canterbury.ac.nz/~m.steel/Non_UC/files/research/conjectures_updated.pdf`.

14 Mayur Thakur and Rahul Tripathi. Linear connectivity problems in directed hypergraphs. *Theoretical Computer Science*, 410(27-29):2592–2618, 2009.