Embedding phylogenetic trees in networks of low treewidth

Leo van Iersel, Mark Jones, Mathias Weller

▶ **To cite this version:**

Leo van Iersel, Mark Jones, Mathias Weller. Embedding phylogenetic trees in networks of low treewidth. Discrete Mathematics and Theoretical Computer Science, 2023, vol. 25:2 (Discrete Algorithms), 10.46298/dmtcs.10116 . hal-04329186

HAL Id: hal-04329186
https://hal.science/hal-04329186

Submitted on 7 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Discrete Mathematics and Theoretical Computer Science . . . . . .
. . . . . **vol. 25:2 #4 (2023)**

Embedding phylogenetic trees in networks of low treewidth

Leo van Iersel[1] Mark Jones[1] Mathias Weller[2]

[1] *Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands*
[2] *CNRS, LIGM (UMR 8049), Université Gustave Eiffel, Champs-sur-Marne, France*

revisions 3rd Oct. 2022, 16th May 2023; accepted 21st June 2023.

Given a rooted, binary phylogenetic network and a rooted, binary phylogenetic tree, can the tree be embedded into the network? This problem, called TREE CONTAINMENT, arises when validating networks constructed by phylogenetic inference methods. We present the first algorithm for (rooted) TREE CONTAINMENT using the treewidth t of the input network N as parameter, showing that the problem can be solved in $2^{O(t^2)} \cdot |N|$ time and space.

Keywords: fixed-parameter tractability, treewidth, phylogenetic tree, phylogenetic network, display graph, tree containment, embedding

1 Introduction

1.1 Background: phylogenetic trees and networks

Phylogenetic trees and networks are graphs used to represent evolutionary relationships. In particular, a *rooted phylogenetic network* is a directed acyclic graph with distinctly labelled leaves, a unique root and no indegree-1 outdegree-1 vertex. Here, we will only consider rooted binary phylogenetic networks, which we will call *networks* for short. The labels of the leaves (indegree-1 outdegree-0 vertices) can, for example, represent a collection of studied biological species, and the network then describes how they evolved from a common ancestor (the root, a unique indegree-0 outdegree-2 vertex). Vertices with indegree 2 and outdegree 1 are called *reticulations* and represent events where lineages combine, for example the emergence of new hybrid species. All other vertices have indegree 1 and outdegree 2. A network without reticulations is a *phylogenetic tree*.

1.2 The TREE CONTAINMENT problem

The evolutionary history of a small unit of hereditary information (for example a gene, a fraction of a gene or (in linguistics) a word) can often be described by a phylogenetic tree. This is because at each reticulation, each unit is inherited from only one parent. Hence, if we trace back the evolutionary history of such a hereditary unit in the network, we see that its phylogenetic tree can be embedded in the network. This raises the fundamental question: given a phylogenetic network and a phylogenetic tree, can the tree be embedded into the network? This is called the TREE CONTAINMENT problem (see Fig. 1). To formalize this problem, we say that a network N *displays* a tree T if some subgraph of N is a subdivision of T.

ISSN 1365–8050 © 2023 by the author(s) Distributed under a Creative Commons Attribution 4.0 International License

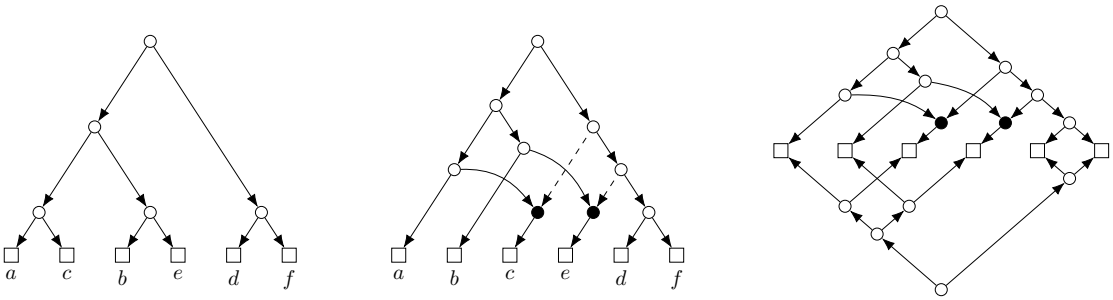

Figure 1: Left: a phylogenetic tree T. **Middle**: a phylogenetic network N displaying T (solid lines indicate an embedding of T; black nodes indicate reticulations). **Right**: the display graph $D(N,T)$ of N and T (see Section 1.4) with the network part drawn on top and the tree part drawn on the bottom. Note that vertices of the display graph are not labelled. In the figure, the leaves (square vertices) are ordered in the same way as in N.

TREE CONTAINMENT (TC)
Input: phylogenetic network N_{IN} and tree T_{IN}, both on the same set of leaf labels
Question: Does N_{IN} display T_{IN}?

1.2.1 Motivation

Apart from being a natural and perhaps one of the most fundamental questions regarding phylogenetic networks, the TREE CONTAINMENT problem has direct applications in phylogenetics. The main application is the validation of phylogenetic network inference methods. After constructing a network, one may want to verify whether it is consistent with the phylogenetic trees. For example, if a heuristic method is used to generate a network for a genomic data set, and tree inference methods are used to generate trees for each gene, then the quality of the produced network can be assessed by computing the fraction of the gene trees that can be embedded into it. In addition, one may want to find the actual embeddings for visualisation purposes and/or to assess the importance of each network arc.

However, our main motivation for studying TREE CONTAINMENT is that it is a first step towards the wider application of treewidth based approaches in phylogenetics (see Sections 1.3 and 1.4). The techniques we develop are not exclusively designed for TREE CONTAINMENT but intended to be useful also for other problems such as NETWORK CONTAINMENT [JM21] and HYBRIDIZATION NUMBER [BS07, vIKL+16, vIKS16, vIL13]. The former is the natural generalization of TREE CONTAINMENT in which we have two networks as input and want to decide whether one can be embedded into the other. It can, in particular, be used to decide whether two networks are isomorphic. In the latter problem, HYBRIDIZATION NUMBER, the input consists of a set of phylogenetic trees, and the aim is to construct a network with at most k reticulations that embeds each of the input trees. Although this will certainly be non-trivial, we expect that at least part of the approach we introduce here can be applied to those and other problems in phylogenetics. We also believe our approach may have application to problems outside of phylogenetics, involving the reconciliation of multiple related graphs.

1.3 Treewidth

The parameters that are most heavily used for phylogenetic network problems (see eg. Section 1.5) are the reticulation number and the level. This is true not only for TREE CONTAINMENT but more generally in the phylogenetic networks literature. Although these parameters are natural, their downside is that they are not necessarily much smaller than the input size. This is why we study a different parameter here.

The *treewidth* of a graph measures its tree-likeness (see definition below), similarly to the reticulation number and level. In that sense, it is also a natural parameter to consider in phylogenetics, where networks are often expected to be reasonably tree-like. A major advantage of treewidth is that it is expected to be much smaller than the reticulation number and level. In particular, there exist classes of networks for which the treewidth is at most a constant factor times the square-root of the level (see [KSW18] for an example). Moreover, a broad range of advanced techniques have been developed for designing FPT algorithms for graph problems when the parameter is the treewidth [Bod88, CNP+11, BCKN15, EGHK21]. For these reasons, the treewidth has recently been studied for phylogenetics problems [JJK+19, KvSW16, KSW18, SW21] and related width parameters have been proposed [BSW20]. However, using treewidth as parameter for phylogenetic problems poses major challenges and, therefore, there are still few algorithms in phylogenetics that use treewidth as parameter (see Section 1.4).

One of the most famous results in treewidth is Courcelle's Theorem [Cou90, ALS91], which states, informally, that graph problems expressible in monadic second-order logic (MSOL) can be solved in linear time on graphs of bounded treewidth. This makes MSOL formulations a powerful tool for establishing FPT results. For practical purposes, it is often preferable to establish a concrete algorithm, since the running times derived via Courcelle's theorem are dominated by a tower of exponentials of size bounded in the treewidth.

It will be convenient to define a *tree decomposition* of a graph $G = (V, E)$ as a rooted tree, where each vertex of the tree is called a *bag* and is assigned a partition (P, S, F) of V, where S is a separator between P and F. We will refer to S as the *present* of the bag. The set P is equal to the union of the presents of all descendant bags (minus the elements of S) and we refer to it as the *past* of the bag. The set $F = V \setminus (S \cup P)$ is referred to as the *future* of the bag. For each edge of the graph, there is at least one bag for which both endpoints of the edge are in the present of the bag. Finally, for each $v \in V$, the bags that have v in the present form a non-empty connected subtree of the tree decomposition. The *width* of a tree decomposition is one less than the maximum size of any bag's present and the *treewidth* $tw(G)$ of a graph G is the minimum width of any tree decomposition of G. The treewidth of a phylogenetic network or other directed graph is the treewidth of the underlying undirected graph.

Our dynamic programming works with *nice* tree decompositions, in which the root is assigned $(V, \varnothing, \varnothing)$ and each bag assigned (P, S, F) has exactly one of four types: *Leaf* bags have $P = S = \varnothing$ (hence $F = V$) and have no child, *Introduce* bags have a single child assigned $(P, S \setminus \{z\}, F \cup \{z\})$ for some $z \in S$, *Forget* bags have a single child assigned $(P \setminus \{z\}, S \cup \{z\}, F)$ for some $z \in P$, and *Join* bags have two children assigned $(L, S, F \cup R)$ and $(R, S, F \cup L)$ respectively, where (L, R) is a partition of P. When the treewidth is bounded by a constant, [Bod96] showed that a minimum-width tree decomposition can be found in linear time and [Klo94] showed that a nice tree decomposition of the same width can be obtained in linear time. Regarding approximation, it is known that, for all graphs G, tree decompositions of width $O(tw(G))$ can be computed in time single-exponential in $tw(G)$ [CFK+15, BDD+16, Kor21] and tree decompositions of width $O(tw(G)\sqrt{\log tw(G)})$ can be computed in polynomial time [FHL08].

1.4 Challenges

One of the main challenges of using treewidth as parameter in phylogenetics is that the central goal in this field is to infer phylogenetic networks and, thus, the network is not known *a priori* so a tree decomposition cannot be constructed easily. A possible strategy to overcome this problem is to work with the *display graph* (see Fig. 1). Consider a problem taking as input a set of trees, such as HYBRIDIZATION NUMBER. Then, the *display graph* of the trees is obtained by taking all trees and identifying leaves with the same label. Now we have a graph in the input and hence we can compute a tree decomposition. Moreover, in some cases, there is a strong relation between the treewidth of the display graph and the treewidth of an optimal network [GKL15, KSW18, JJK+19].

A few instances of exploiting (tree decompositions of) the display graph of input networks for algorithm design have been published. Famously, Bryant and Lagergren [BL06] designed MSOL formulations solving the TREE CONSISTENCY problem on display graphs, which have been improved by a concrete dynamic programming on a given tree decomposition of the display graph [BPSC17]. Kelk et al. [KvSW16] also developed MSOL formulations on display graphs for multiple incongruence measures on trees, based on so-called "agreement forests".

For the TREE CONTAINMENT problem, MSOL formulations acting on the display graph have been used to prove fixed-parameter tractability with respect to the treewidth [JJK+19]. Analogously to the work of Baste et al. [BPSC17] for TREE CONSISTENCY, we develop in this manuscript a concrete dynamic programming algorithm for TREE CONTAINMENT acting on display graphs.

TREE CONTAINMENT is conceptually similar to HYBRIDIZATION NUMBER in the sense that the main challenge is to decide which tree vertices correspond to which vertices of the other trees (for HYBRIDIZATION NUMBER) or network vertices (for TREE CONTAINMENT). However, HYBRIDIZATION NUMBER is even more challenging since the network may contain vertices that do not correspond to any input vertex [vIKL+16]. Therefore, TREE CONTAINMENT is a natural first problem to develop techniques for, aiming at extending them to HYBRIDIZATION NUMBER and other problems in phylogenetics in the long run.

That being said, solving TREE CONTAINMENT parameterized by treewidth poses major challenges itself. Even though the general idea of dynamic programming on a tree decomposition is clear, its concrete use for TREE CONTAINMENT is severely complicated by the fact that the tree decomposition does not know the correspondence between tree vertices and network vertices. For example, when considering a certain bag of the tree decomposition, a tree vertex that is in the present of that bag may have to be embedded into a network vertex that is in the past or in the future. It may also be necessary to map vertices from the future of the tree to the past of the network and vice versa. Therefore, it will not be possible to "forget the past" and "not worry about the future". In particular, this makes it much more challenging to bound the number of possible assignments for a given bag. We will do this by bounding the number of "time-travelling" vertices by a function of the treewidth. We will describe these challenges in more detail in Section 2.2.

1.5 Previous work

TREE CONTAINMENT was shown to be NP-hard [KNTX08], even for tree-sibling, time-consistent, regular networks [ISS10]. On the positive side, polynomial-time algorithms were found for other restricted classes, including tree-child networks [GGL+15, GGL+18, Gun18, GYZ19, ISS10, Wel18]. The first non-trivial FPT algorithm for TREE CONTAINMENT on general networks had running time $O(2^{k/2}n^2)$, where the parameter k is the number of reticulations in the network [KNTX08]. Another algorithm was proposed by [GLZ16] with the same parameter, but it is only shown to be FPT for a restricted class of networks. Since

the problem can be split into independent subproblems at non-leaf cut-edges [ISS10], the parameterization can be improved to the largest number of reticulations in any biconnected component (block), also called the *level* of the network. Further improving the parameterization, the maximum number t^* of "unstable component-roots" per biconnected component was considered and an algorithm (working also in the non-binary case) was found with running time $O(3^{t^*}|N||T|)$ [Wel18]. Herein, a parameterization "improves" over another if the first is provably smaller than (a function of) the second in any input network.

Several generalizations and variants of the TREE CONTAINMENT problem have been studied. The more general NETWORK CONTAINMENT problem asks to embed a *network* in another and has been shown to be solvable in polynomial time on a restricted network class [JM21]. When allowing multifurcations and non-binary reticulations, two variants of TREE CONTAINMENT have been considered: In the FIRM version, each non-binary node ("polytomy") of the tree has to be embedded in a polytomy of the network whereas, in the SOFT version, polytomies may be "resolved" into binary subtrees in any way [BMW18]. Finally, the unrooted version of TREE CONTAINMENT was also shown to be NP-hard but fixed-parameter tractable when the parameter is the reticulation number (the number of edges that need to be deleted from the network to obtain a tree) [IKS+18]. While this version of the problem is also known to be fixed-parameter tractable with respect to the treewidth of the network [JJK+19], the work does not explicitly describe an algorithm and the implied running time depends on Courcelle's theorem [Cou97] which makes practical implementation virtually impossible.

Since the notion of "display" closely resembles that of "topological minor" (with the added constraint that the embedding must respect leaf-labels), TREE CONTAINMENT can be understood as a special case of a variant of the well-known TOPOLOGICAL MINOR CONTAINMENT (TMC) problem for directed graphs. TMC is known to be NP-complete in general by reduction from HAMILTONIAN CYCLE and previous algorithmic results focus on the undirected variant, parameterized by the *size* h of the sought topological minor H (corresponding to the input tree for TREE CONTAINMENT). In particular, undirected TMC can be solved in $f(h)n^{O(1)}$ time [GKMW11, FLP+20]. However, the dependency of the function f on h makes such algorithms impractical for all but small values of h. By contrast, in TREE CONTAINMENT the input tree may be assumed to typically have a size comparable to the overall input size. In the directed case, even the definition of "topological minor" has been contested [GHK+16] and we are aware of little to no algorithmic results. In TREE CONTAINMENT, part of the embedding of the host tree in the guest network is fixed by the leaf-labeling. If the node-mapping is fixed for *all* nodes of the host, then directed TMC generalizes the DISJOINT PATHS problem [FHW80], which is NP-complete for 2 paths or in case the host network is acyclic. Indeed, one can show TREE CONTAINMENT to be NP-hard in a similar fashion [KNTX08].

1.6 Our contribution

In this paper, we present an FPT-algorithm for TREE CONTAINMENT parameterized by the treewidth of the input network. Our algorithm is one of the first (constructive) FPT-algorithms for a problem in phylogenetics parameterized by treewidth. We believe that this is an important development as the treewidth can be much smaller than other parameters such as reticulation number and level which are easier to work with. We see this algorithm as an important step towards the wide application of treewidth-based methods in phylogenetics.

2 Preliminaries

2.1 Reformulating the problem

A key concept throughout this paper will be *display graphs* [BL06], which are the graphs formed from the union of a phylogenetic tree and a phylogenetic network by identifying leaves with the same labels. Throughout this paper we will let N_{IN} and T_{IN} denote the respective input network and tree in our instance of TREE CONTAINMENT. The main object of study will be the "display graph" of N_{IN} and T_{IN}. For the purposes of our dynamic programming algorithm, we will often consider graphs that are not exactly this display graph, but may be thought of as roughly corresponding to subgraphs of it (though they are not exactly subgraphs; see Section 3.1). In order to incorporate such graphs as well, we will define display graphs in a slightly more general way than that usually found in the literature. In particular, we allow for the two "sides" of a display graph to be disconnected, and for some leaves to belong to one side but not the other.

Definition 1 (display graph)**.** *A* display graph *is a directed acyclic graph $D = (V, A)$, with specified subsets $V_T, V_N \subseteq V$ such that $V_T \cup V_N = V$, satisfying the following properties:*

- *The graph $T := D[V_T]$ is an out-forest;*
- *Every vertex has in- and out-degree at most 2 and total degree at most 3;*
- *Any vertex in $V_N \cap V_T$ has out-degree 0 and in-degree at most 1 in both T and $N := D[V_N]$.*

Herein, we call T the tree side *and N the* network side *of D and we will use the term $D(N, T)$ to denote a display graph with network side N and tree side T.*

Given a phylogenetic network N_{IN} and phylogenetic tree T_{IN} with the same leaf-label set, we define $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ to be the display graph formed by taking the disjoint union of N_{IN} and T_{IN} and identifying pairs of leaves that have the same label. We note that, while the leaves of N_{IN} and T_{IN} were originally labelled, this labelling does not appear in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Labels are used to construct $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, but in the rest of the paper we will not need to consider them. Indeed, such labels are relevant to the TREE CONTAINMENT problem only insofar as they establish a relation between the leaves of T_{IN} and N_{IN}, and this relation is now captured by the structure of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

We now reformulate the TREE CONTAINMENT problem in terms of an *embedding function* on a display graph. Unlike the standard definition of an embedding function (see, e.g., [ISS10]), which is defined for a phylogenetic network N and tree T, our definition of an embedding function applies directly to the display graph $D(N, T)$. Because of our more general definition of display graphs, our definition of an embedding function will also be more general than that found in the literature. The key idea of an embedding function remains the same, however: it shows how a subdivision of T may be viewed as a subgraph of N.

Definition 2 (embedding function)**.** *Let D be a display graph with network side N and tree side T, and let $\mathcal{P}(N)$ denote the set of all directed paths in N. An* embedding function *on D is a function $\phi : V(T) \cup A(T) \to V(N) \cup \mathcal{P}(N)$ such that:*

(a) for each $u \in V(T)$, $\phi(u) \in V(N)$ and, for each $uv \in A(T)$, $\phi(uv)$ is a directed $\phi(u)$-$\phi(v)$-path in N;
(b) for any distinct $u, v \in V(T)$, $\phi(u) \neq \phi(v)$;
(c) for any $u \in V(T) \cap V(N)$, $\phi(u) = u$;
(d) the paths $\{\phi(uv) \mid uv \in A(T)\}$ are arc-disjoint;
(e) for any distinct $p, q \in A(T)$, $\phi(p)$ and $\phi(q)$ share a vertex z only if p and q share a vertex w with $z = \phi(w)$;

Note that the standard definition of an embedding of a phylogenetic tree T into a phylogenetic network N (see e.g. [ISS10]) coincides with the definition of an embedding function on $D(N, T)$. Property (e) ensures

that, while the embeddings of arcs uv, vw_1, vw_2 can all meet at $\phi(v)$, the embeddings of different tree arcs cannot otherwise meet. (In particular, the path $\phi(uv)$ cannot end at a reticulation that is also an internal vertex of $\phi(u'v')$, something that is otherwise allowed by properties (a)–(d).)

Lemma 1. *A phylogenetic network N_{IN} displays a phylogenetic tree T_{IN} if and only if there is an embedding function on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.*

Proof: Suppose first that N_{IN} displays T_{IN}, that is, there is a subgraph T'_{IN} of N_{IN} that is a subdivision of T_{IN}. Then, every vertex of T_{IN} corresponds to a vertex of T'_{IN} and every arc uv in T_{IN} corresponds to a directed path in T'_{IN} between the vertices corresponding to u and v. Let ϕ denote this correspondence relation. Then, (c) of Definition 2 follows from the definition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ while the other properties follow from ϕ being an isomorphism (of a subdivision of T_{IN} into T'_{IN}).

Conversely, suppose that there is an embedding function ϕ on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ and let T'_{IN} be the subgraph formed by the arcs of N_{IN} that are part of some path $\phi(uv)$ for an arc uv in T. Then, it can be verified that T'_{IN} is a subdivision of T_{IN}. □

In light of Lemma 1, we may henceforth view TREE CONTAINMENT as the following problem:

TREE CONTAINMENT (TC)
Input: phylogenetic network N_{IN} and phylogenetic tree T_{IN} with the same leaf-label set
Task: Find an embedding function on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

2.2 Overview of our approach

We study TREE CONTAINMENT parameterized by the treewidth of the input network N_{IN}. A key tool will be the following theorem from [JJK+19]. In this theorem, the display graph $D_{\text{IN}}(N_u, T_u)$ for unrooted N_u and T_u is defined analogously to $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ for rooted $N_{\text{IN}}, T_{\text{IN}}$ – that is, it is the (undirected) graph derived from the disjoint union of N_u and T_u by identifying leaves with the same label.

Theorem 1 ([JJK+19])**.** *Let N_u and T_u be an unrooted binary phylogenetic network and tree, respectively, with the same leaf-label set. If N_u displays T_u then $tw(D_{\text{IN}}(N_u, T_u)) \leq 2tw(N_u) + 1$.*

By Theorem 1, we suppose that the display graph $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ has treewidth at most $2k + 1$, where k is the treewidth of the underlying undirected graph N_u of N_{IN}, as otherwise $tw(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) = tw(D_{\text{IN}}(N_u, T_u)) > 2k + 1$ and N_u does not display the unrooted version T_u of T_{IN}, implying that N_{IN} does not display T_{IN}.

As is often the case for treewidth parameterizations, we will proceed via a dynamic programming on a tree decomposition, in this case a tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Recall that we view a bag (P, S, F) in the tree decomposition as partitioning the vertices of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ into past, present and future. A typical dynamic programming approach is to store, for each bag, some set of information about the present, while forgetting most information about the past, and not yet caring about what happens in the future. The resulting information is stored in a "signature", and the algorithm works by calculating which signatures are possible on each bag, in a bottom-up manner. This approach is complicated by the fact that the sought-for embedding of T_{IN} into N_{IN} may not map the past/present/future of T_{IN} into the past/present/future (respectively) of N_{IN}. Vertices from the past of T_{IN} may be embedded in the future of N_{IN}, or vice versa. Thus, we have to store more information than we might at first think. In particular, it is not enough to store information about which present vertices of T_{IN} are embedded in which present vertices of N_{IN} (indeed, depending on the bag, it may be that none of them are). As such, our notion of

a "signature" has to track how vertices from the past of T_{IN} are embedded in the present and future of N_{IN}, and which vertices from the past of N_{IN} contain vertices from the present or future of T_{IN}. Vertices of the past which are mapped to vertices of the past, on the other hand, can mostly be forgotten about.

2.3 An informal guide to (compact) signatures

Roughly speaking, a *signature* σ for a bag (P, S, F) in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ consists of the following items (see Fig. 3 for an example):

1. a display graph $D(N, T)$, some of whose vertices correspond (isomorphically) to $S \subseteq V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$, and the rest of which are labeled PAST or FUTURE (which we may think of as vertices corresponding to some vertex of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ in P or F, respectively). We use a function ι on $V(D(N, T))$ to capture both this correspondence and labelling, where ι maps each vertex to an element of S or a label from $\{\text{PAST}, \text{FUTURE}\}$. We emphasize here that $D(N, T)$ is distinct from the input display graph $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

2. an embedding ϕ of T in N such that, for no arc uv of T, all of $V(\phi(uv)) \cup \{u, v\}$ have the same label $y \in \{\text{PAST}, \text{FUTURE}\}$ under ι.

Signatures may be seen as "partial embedding functions" on parts of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ in a straightforward way. In particular, we call σ *valid* for (P, S, F) if, roughly speaking, ϕ corresponds (via ι) to something that can be extended to an embedding function on the subgraph of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ induced by the vertices $P \cup S$ introduced below (P, S, F). In our dynamic programming algorithm, we build valid signatures for a bag x from valid signatures of the child bag(s) of x (in particular, validity of a signature for x is characterized by the validity of certain signatures for the child bag(s)).

Since the definition of a signature does not guarantee any bound on the number of vertices labeled PAST or FUTURE, iterating over all signatures for a bag (P, S, F) (in order to check their validity) exceeds FPT time. Therefore we will instead consider "compact" signatures, whose number and size are bounded in the width $|S|$ of the bag (P, S, F). If $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ admits an embedding function ϕ^*, then a compact signature corresponding to this embedding function exists. In the following, we informally describe the compaction process for this hypothetical solution ϕ^*, thus giving a rough idea of the definition of a "compact" signature. At all times, the (tentative) signature will contain a display graph $D(N, T)$ (initially $D(N, T) = D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$), and an embedding function of T into N (initially ϕ^*). For a more complete description of our approach, see Section 3 and, for an illustration, see Fig. 2.

Step 1 After initialization with ϕ^*, we assign a label FUTURE to all vertices of F, and a label PAST to all vertices in P (Observe that no vertex labeled PAST will be adjacent to a vertex labeled FUTURE, since S separates P from F in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$). Then, we "forget" which vertices of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ the vertices labelled PAST or FUTURE correspond to. Our preliminary signature now contains (1) a display graph $D(N, T)$ whose vertices are either labelled FUTURE or PAST or correspond (isomorphically) to vertices in $S \subseteq V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ (we refer the reader to Section 3.1 for a more formal description), as well as (2) an embedding function for $D(N, T)$ into N.

Step 2 We now simplify the structure of the preliminary signature. The main idea is that, if a is an arc of T with both endpoints labelled PAST and all vertices in the path $\phi(a)$ are also labelled PAST, then we can safely forget a and all the arcs in $\phi(a)$. Intuitively, the information that a will be embedded in $\phi(a)$ does not have any effect on the possible solutions one could construct on the part of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ that is "above" the bag (P, S, F). Similarly, we can forget any arc a of T whose endpoints, as well

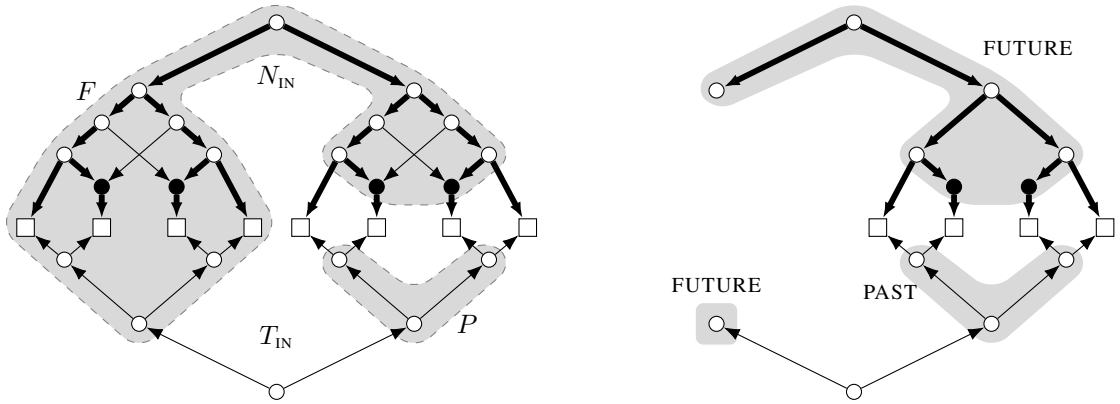

Figure 2: Left: An example of a display graph $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ for which N_{IN} displays T_{IN} as witnessed by the embedding function ϕ that is indicated by bold edges. Highlighting with dashed border represents the sets P and F, for some bag (P, S, F) in a tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. **Right**: A representation of the (compact) signature for (P, S, F) derived from this solution. Vertices labelled PAST or FUTURE are highlighted in gray without border.

as every vertex in $\phi(a)$, are assigned the label FUTURE. Intuitively, this is because this information should have no bearing on whether a solution exists with this signature for $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ restricted to $P \cup S$. In a similar way, we forget any vertex $u \in V(T)$ and its embedding $\phi(u)$ if they are assigned the same label, provided that all their incident arcs can also be forgotten. We will call the vertices and arcs fulfilling these conditions "redundant" and we remove them from our tentative signature. We can also safely delete the vertices and arcs of N that are labelled $y \in \{\text{PAST}, \text{FUTURE}\}$ but are not part of the image of ϕ. As a result, we now have that for any remaining vertex $u \in T$, either one of $\{u, \phi(u)\}$ is labelled PAST and the other labelled FUTURE, or some vertex element of S must appear as either one of $\{u, \phi(u)\}$, a neighbor of u, or a vertex in the path $\phi(a)$ for an incident arc a of u. Thus, we have "forgotten" all the aspects of the embedding except those that involve vertices from the present in some way, or those where the embedding "time-travels" between the past and future (see Section 3.3 for a more formal description of this process).

Step 3 Finally, we may end up with long paths of vertices with in-degree and out-degree 1 that are labelled PAST or FUTURE in N (for example, if u and v are labelled PAST, then $\phi(uv)$ may be a long path in N with all vertices labelled FUTURE). Such long paths do not contain any useful information to us, we therefore compress these by suppressing vertices with in-degree and out-degree 1 (This gives the *compact signature*, see Section 3.8).

2.4 Bounding the number of signatures

We now outline the main arguments for why the number of possible (compact) signatures for a given bag (P, S, F) can be bounded in $|S|$. Such a bound on the number of signatures ensures that the running time of the algorithm is FPT, because the number of calculations required for each bag is bounded by a function of the treewidth.

The main challenge is to bound the size of the display graph $D(N, T)$ in a given signature for (P, S, F). Once such a bound is achieved, this immediately implies upper bounds (albeit quite large) for the number

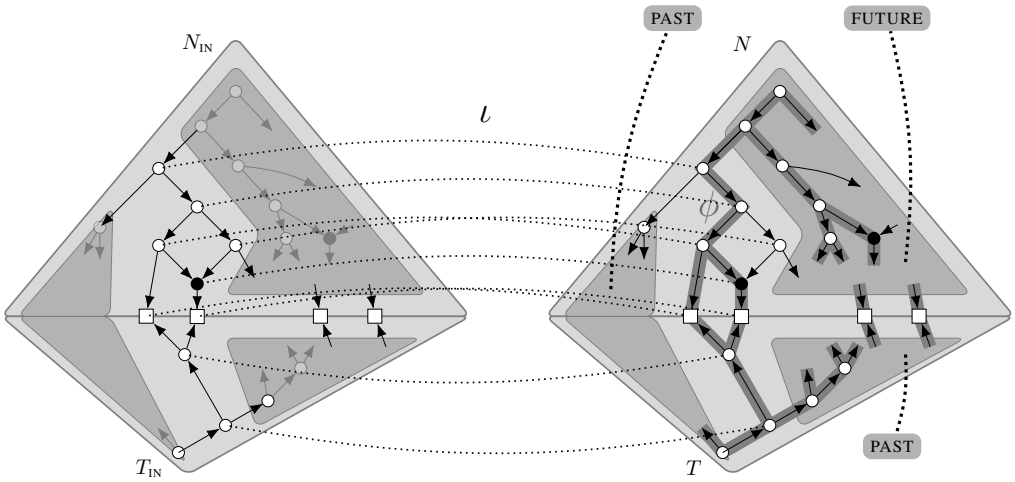

Figure 3: Example of a signature of a bag (P, S, F). The S-part of $D(N_{\text{IN}}, T_{\text{IN}})$ is solid while the non-S part is faded. The embedding ϕ (right, indicated with gray edge-highlight) maps T into N. The dotted arcs labelled ι show the isomorphism between part of $D(N, T)$ and $S \subseteq V(D(N_{\text{IN}}, T_{\text{IN}}))$. Note that the part of $D(N, T)$ that is not mapped to S is not necessarily isomorphic to anything in $D(N_{\text{IN}}, T_{\text{IN}})$.

of possible display graphs and the number of possible embeddings, and hence on the number of possible signatures. We will focus here on bounding the size of the tree part T. Once a bound is found for $|T|$ it is relatively straightforward to use that to give a bound on $|N|$ (because the arcs of N that are not used by the embedding of T into N are automatically deleted, unless they are themselves incident to a vertex in S, and because isolated vertices are deleted and long paths suppressed).

It can be seen that a vertex $u \in V(T)$ is redundant (and so would be deleted from the signature) unless one of the following properties holds: (1) $u \in S$, (2) $\phi(u) \in S$, (3) u is incident to an element of S (4) for some arc a incident to u, the path $\phi(a)$ contains a vertex from S or (5) u and $\phi(u)$ have different labels from $\{\text{PAST}, \text{FUTURE}\}$. Essentially if none of (1)–(4) holds, then all the vertices mentioned in those properties have the same label as either u or $\phi(u)$, using the fact that S separates the vertices labelled PAST from the vertices labelled FUTURE. If u and $\phi(u)$ have the same label, then all these vertices have the same label, which is enough to show that u is redundant. It remains to bound the number of vertices satisfying one of these properties. For the first four properties, it is straightforward to find a bound in terms of $|S|$. The vertices satisfying the final property are "time-travelling" (in the sense that either u is labelled PAST and $\phi(u)$ FUTURE, or u is labelled FUTURE and $\phi(u)$ PAST). Because of the bounds on the other types of vertices, it is sufficient to provide a bound on the number of *lowest* time-travelling vertices in T.

To see the intuition why this bound should hold: consider some full solution on the original input, i.e. an embedding function on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, and suppose $u \in V(T_{\text{IN}})$ is a lowest tree vertex for which $u \in P, \phi(u) \in F$ (thus in the corresponding signature, u has label PAST and $\phi(u)$ has label FUTURE). Let $x \in V(N_{\text{IN}}) \cap V(T_{\text{IN}})$ be some leaf descendant of u. Then there is path in T_{IN} from u to x, and a path in N_{IN} from $\phi(u)$ to $\phi(x) = x$. Thus $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ has an (undirected) path from u to $\phi(u)$. As this is a path between a vertex in P and a vertex in F, some vertex on this path must be in S (since S separates P from F). Such a path must exist for every lowest time-travelling vertex u, and these paths are distinct. The

existence of these paths can then be used to bound the number of lowest time-travelling vertices.

3 Filling in the details

3.1 Tracking the identity of vertices

In the following, consider a bag (P, S, F) and recall that the sought solution may embed parts of the past of T in the future of N or vice versa. Thus, as previously indicated, a signature for (P, S, F) will have to track information about more vertices than those in S. This leads to the following complexities in how we talk about the identity of vertices in signatures.

For the display graph $D(N, T)$ in a signature for (P, S, F), we want some vertices to correspond to specific vertices in $S \subseteq V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ while, for other vertices we want to express the fact that they correspond to a vertex of P or F, without specifying which one. Moreover, as our dynamic programming proceeds up the tree decomposition and we move from one signature to another, vertices will change between these two states. When we move from a child bag (P_c, S_c, F_c) to a parent bag (P_p, S_p, F_p) and a vertex x of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ "leaves the bag" (that is, $x \in S_c \setminus S_p$) then we want to "forget" that a vertex v corresponds to x (instead labelling v PAST), while still tracking its adjacencies and embedding information. Similarly, a vertex labelled FUTURE may, at some point, come to correspond to a particular vertex in S.

In order to keep this information straight, we have to be careful in how we talk about the identity of these vertices. In particular, the vertices of the display graph $D(N, T)$ are different from those in $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) = P \uplus S \uplus F$, but the signature carries a "correspondence" function ι that, on some subgraph of $D(N, T)$, acts as an isomorphism into the "S-part" of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ while, on other vertices, acting as a labelling that assigns some vertices the label PAST and others the label FUTURE. As such, we refer to ι as an "isolabelling".

In the next definition, we consider an arbitrary set $\mathcal{Y}$ of labels, rather than the set $\{\text{PAST}, \text{FUTURE}\}$. While we will often have $\mathcal{Y} = \{\text{PAST}, \text{FUTURE}\}$, we also use isolabelings in other contexts besides signatures.

Definition 3 (isolabelling)**.** *Let $\mathcal{Y}$ be a set of labels and let $S \subseteq V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$. An $(S, \mathcal{Y})$-*isolabelling *on a display graph $D(N, T)$ is a function $\iota : V(D(N, T)) \to S \cup \mathcal{Y}$ such that S is a subset of the image of ι (ι is "surjective onto S")* [(i)] *and, for any $u, v \in V(D(N, T))$ with $\iota(u), \iota(v) \in S$:*
(a) $\iota(u) \in V(N_{\text{IN}})$ only if $u \in V(N)$ and $\iota(u) \in V(T_{\text{IN}})$ only if $u \in V(T)$,
(b) $\iota(u) = \iota(v)$ only if $u = v$, and
(c) the arc uv is in $D(N, T)$ if and only if $\iota(u)\iota(v)$ is in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

3.2 Signatures and containment structures

Before formally defining a signature for a bag (P, S, F), let us remark that, later in the paper, we will also discuss constructs which are similar in construction to signatures, but with slightly different vertex and label sets ("partial solutions" and "reconciliations"). For this reason, we define a more general structure, called an $(S, \mathcal{Y})$-containment structure, that encapsulates all these notions. In the definition that follows, a signature for a bag (P, S, F) corresponds to an $(S, \mathcal{Y})$-containment structure, where S is the present of the bag (P, S, F) and $\mathcal{Y}$ is the label set $\{\text{PAST}, \text{FUTURE}\}$. For an intuitive understanding of a containment structure, the most important features are the display graph, embedding function and isolabelling.

[(i)] Every $v \in S$ has a $u \in V(D(N, T))$ with $\iota(u) = v$, but not every $y \in \mathcal{Y}$ may have a $u \in V(D(N, T))$ with $\iota(u) = y$.

Definition 4 (containment structure)**.** *Let $\mathcal{Y}$ be a set of labels and let $S \subseteq V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$. An $(S, \mathcal{Y})$-*containment structure *is a tuple $(D(N,T), \phi, \iota)$ consisting of*

- *a display graph $D(N,T)$,*
- *an embedding function ϕ on $D(N,T)$, and*
- *an $(S, \mathcal{Y})$-isolabelling $\iota : V(D(N,T)) \to S \cup \mathcal{Y}$.*

In addition, each vertex u of $D(N,T)$ should satisfy the following properties:
(a) if $\iota(u) \in S$, then u has the same in- and out-degree in $D(N,T)$ as $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.
(b) if $u \in T$ and $\iota(u) \neq \iota(\phi(u))$, then u has 2 out-arcs in $D(N,T)$.

See Fig. 3 for an example of an $(S, \{\text{PAST}, \text{FUTURE}\})$-containment structure $(D(N,T), \phi, \iota)$. The last two properties of Definition 4 are used for bookkeeping, and to help bound the size of the display graph.

Definition 5 (signature)**.** *Let (P, S, F) be a bag in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Then any $(S, \{\text{PAST}, \text{FUTURE}\})$-containment structure is called a* signature for (P, S, F).

Through most of the paper, $\mathcal{Y} = \{\text{PAST}, \text{FUTURE}\}$ and S may be referred to as the "present" of the bag (P, S, F). However, our setup also allows us to talk about "partial solutions", where S is replaced by $P \cup S$ (essentially merging the past and the present) and only the label set $\mathcal{Y} = \{\text{FUTURE}\}$ (corresponding to vertices in F) is used. In some instances, we will use additional auxiliary labels; in particular when considering Join bags in Section 3.6, we use the labels LEFT and RIGHT to distinguish between the pasts of different bags. Finally, Definition 4 also allows capturing a solution for an instance of TREE CONTAINMENT. Indeed, if we replace the set S with $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and require that no vertex is mapped to a label of $\mathcal{Y}$, then $D(N,T)$ is isomorphic to $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ and the embedding function ϕ of T into N also describes an embedding of T_{IN} into N_{IN}.

Lemma 2. *$(N_{\text{IN}}, T_{\text{IN}})$ is a* YES*-instance of* TREE CONTAINMENT *if and only if there is a $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \mathcal{Y})$-containment structure $(D(N,T), \phi, \iota)$ with $\iota^{-1}(\mathcal{Y}) = \varnothing$.*

Proof: First, suppose $(D(N,T), \phi, \iota)$ is a $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \mathcal{Y})$-containment structure with $\iota^{-1}(\mathcal{Y}) = \varnothing$. Then, for every vertex u of $D(N,T)$, $\iota(u)$ is a vertex of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Moreover, for every vertex v of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, there is $u \in V(D(N,T))$ such that $v = \iota(u)$ (see Definition 4). Thus, ι is a bijective function between $V(D(N,T))$ and $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and, by Definition 3(c), uv is an arc in $D(N,T)$ if and only if $\iota(u)\iota(v)$ is an arc in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Thus, ι is an isomorphism, implying that $D(N,T)$ is isomorphic to $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Moreover N is isomorphic to N_{IN} and T is isomorphic to T_{IN} (as ι maps vertices of N to vertices of N_{IN} and vertices of T to vertices of T_{IN} by Definition 3(a)). As ϕ is an embedding function on $D(N,T)$, combining ι with ϕ yields an embedding function on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

Conversely, suppose that there is an embedding function ϕ on $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Let ι be the identity function on $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and note that, by Definition 3, ι is a $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \mathcal{Y})$-isolabelling on $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and $\iota^{-1}(\mathcal{Y}) = \varnothing$. Then, $(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}), \phi, \iota)$ satisfies the first three conditions of an $(S, \mathcal{Y})$-containment structure. The remaining properties, concerning the degrees of vertices, follow from the fact that N_{IN} and T_{IN} are binary. In particular, for any vertex u in T_{IN} with $\iota(u) \neq \iota(\phi(u))$, it holds that $u \neq \phi(u)$ and so u is not a leaf of T_{IN} and, thus, has out-degree 2 in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. □

3.3 Formally defining the restriction

While Section 2.3 describes in broad strokes how a signature for a bag (P, S, F) could be derived from a solution for an instance of TREE CONTAINMENT, we now make this process more precise. Recall that we

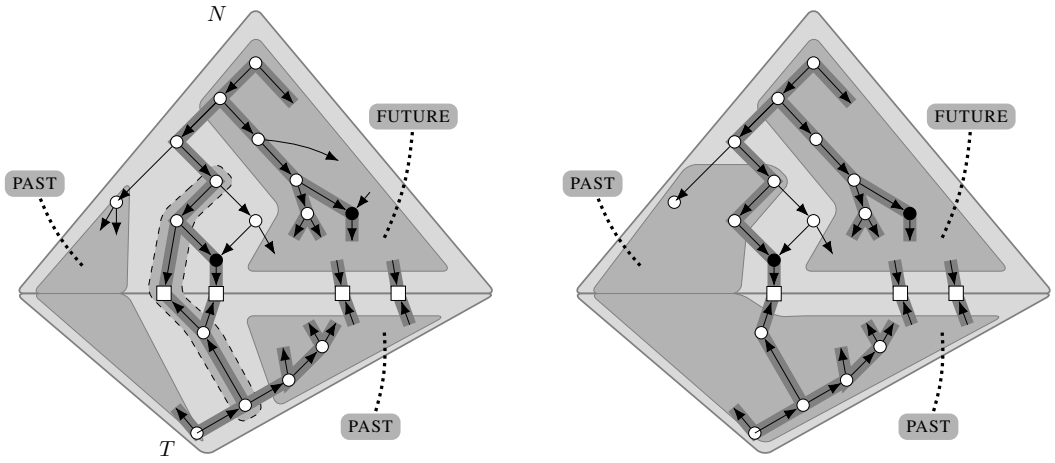

Figure 4: Illustration of Definition 7 (except non-$\mathcal{Y}$ parts of ι and ι'). **Left**: The example containment structure ψ of Fig. 3. The dashed area indicates S' and g: for all u in the dashed area, $g(\iota(u)) = \text{PAST}$ while $\iota(u) \notin \{\text{PAST}, \text{FUTURE}\}$. **Right**: The g-restriction of ψ. In accordance with Definition 7, note how (a) arcs in the "PAST"-area of N that are not mapped to by ϕ disappear in N' while arcs that are mapped by ϕ persist, (b) many of the arcs in T are in the "PAST"-area, but embedded by ϕ into paths of N that live (at least partially) in the "FUTURE"-area of N and, therefore, also persist, and (c) a common leaf of T and N has been removed since all its incoming arcs have been deleted. Finally, as will be the case in the dynamic programming later on, the "PAST"- and "FUTURE"-areas never touch in $D(N', T')$.

first relabelled the vertices in P and F by PAST and FUTURE, respectively, and then, "redundant" parts of the display graph were removed and the remaining long paths contracted. Exactly which arcs and vertices should be removed is made precise in the notion of *redundancy* defined below.

Definition 6 (redundant)**.** *Let $\chi := (D(N,T), \phi, \iota)$ be an $(S, \mathcal{Y})$-containment structure, and let $y \in \mathcal{Y}$. Then, we define the* y-redundant *arcs and vertices of $D(N,T)$ as follows:*

- *A tree arc $uv \in A(T)$ is y-redundant if $\iota(u) = \iota(v) = y$ and $\iota(v') = y$ for all vertices v' in the path $\phi(uv)$.*
- *A network arc $u'v' \in A(N)$ is y-redundant if $\iota(u') = \iota(v') = y$ and, if $u'v'$ is in the path $\phi(uv)$ for some tree arc $uv \in A(T)$, uv is y-redundant.*
- *A tree vertex v in $V(T)$ is y-redundant if $\iota(v) = \iota(\phi(v)) = y$, and v and $\phi(v)$ are incident only to y-redundant arcs in $D(N,T)$.*
- *A network vertex $v' \in V(N)$ is y-redundant if $\iota(v') = y$ and v' is incident only to y-redundant arcs and, if $v' = \phi(v)$ for some $v \in V(T)$, v is y-redundant.*

We say that an arc or vertex of $D(N,T)$ is redundant *if it is y-redundant for some $y \in \mathcal{Y}$. When it is important to specify the containment structure χ, we say an arc or vertex is* y-redundant with respect to χ.

Just as we derived a signature from a solution in Section 2.3 by restricting our attention to a subset of vertices (in that case, the set S), we can restrict any $(S, \mathcal{Y})$-containment structure to an $(S', \mathcal{Y})$-containment structure for any $S' \subseteq S$. This will be a useful tool for deriving signatures for one bag from signatures for another bag, as well as characterizing the "validity" of a signature.

Definition 7 (restriction, see Fig. 4)**.** *Let $\chi = (D(N,T), \phi, \iota)$ be an $(S, \mathcal{Y})$-containment structure, let $S' \subseteq S$, and let $g : S \cup \mathcal{Y} \to S' \cup \mathcal{Y}$ be some function such that for all $v \in S \cup \mathcal{Y}$, $g(v) = v$ if $v \in S'$ and $g(v) \in \mathcal{Y}$ otherwise. Then, we call g a* restriction function*. Further, the g-*restriction *of χ is the tuple $(D(N',T'), \phi', \iota')$ constructed as follows:*

With $\iota_g(u)$ abbreviating $g(\iota(u))$ for all $u \in V(D(N,T))$, let $D(N',T')$ be the display graph derived from $D(N,T)$ by replacing the isolabelling ι with ι_g, and then exhaustively deleting redundant arcs and vertices (with respect to $(D(N,T), \phi, \iota_g)$) from $D(N,T)$. Finally, we define ι' and ϕ' as the respective restrictions of ι_g to $D(N',T')$ and ϕ to T'.

We will sometimes write $(S_1 \to y_1, \dots, S_j \to y_j)$-restriction instead of g-restriction, where $y_1, \dots y_j$ are labels in $\mathcal{Y}$ and $g(u) := y_i$ for all $u \in S_i$, and $g(u) := u$ for all other u. For example, we may write $(P \to \text{PAST})$-restriction when g is the function that maps all vertices in P to the label PAST, and leaves other vertices unchanged. If any S_i is a singleton set $\{z\}$, we permit ourselves to write $z \to y_i$ instead of $\{z\} \to y_i$. In particular, we will see $(\text{RIGHT} \to \text{FUTURE})$-restrictions, where the label RIGHT is mapped to the label FUTURE. Such restrictions may sometimes be used to "merge" labels. Note that Steps 1 and 2 of the process described in Section 2.3 is (roughly analogous to) the process of constructing the $(P \to \text{PAST}, F \to \text{FUTURE})$-restriction of some $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}), \varnothing)$-containment structure corresponding to a solution.

Before proving results relating to g-restrictions, we make the following observations about g-restrictions and redundant arcs and vertices, which are straightforward consequences of Definition 6 and, thus, we omit their proofs.

Observation 1. *Any arc or vertex a of $D(N,T)$ is y-redundant with respect to $(D(N,T), \phi, \iota)$ if and only if $\iota(Q_a) = \{y\}$ for a set of vertices Q_a that depends only on $D(N,T)$ and ϕ:*

- *If a is a tree arc, then $Q_a = V(a) \cup V(\phi(a))$.*
- *If a is a network arc, then $Q_a = Q_{a'}$ if Q_a is part of a path $\phi(a')$, otherwise $Q_a = V(a)$.*
- *If a is a tree vertex, then $Q_a = a \cup \phi(a) \cup \bigcup_{uv \in A(D(N,T)) : a \in \{u,v\}} Q_{uv}$.*
- *If a is a network vertex and $a = \phi(u')$, then $Q_a = Q_{u'} \cup \bigcup_{uv \in A(D(N,T)) : a \in \{u,v\}} Q_{uv}$, otherwise $Q_a = a \cup \bigcup_{uv \in A(D(N,T)) : a \in \{u,v\}} Q_{uv}$.*

Observation 2. *If a tree arc uv is y-redundant then so is every arc in $\phi(uv)$. A tree vertex u is y-redundant if and only if $\phi(u)$ is y-redundant.*

Lemma 3. *Let χ be an $(S, \mathcal{Y})$-containment structure, let $S' \subseteq S$, and let χ' be the g-restriction of χ for some restriction function $g : S \cup \mathcal{Y} \to S' \cup \mathcal{Y}$. Then, χ' is an $(S', \mathcal{Y})$-containment structure.*

Proof: To show that $\chi = (D(N,T), \phi, \iota)$ being an $(S, \mathcal{Y})$-containment structure implies $\chi' = (D(N',T'), \phi', \iota')$ being an $(S', \mathcal{Y})$-containment structure, we verify the five conditions of Definition 4 individually.

$D(N',T')$ is a display graph. As $D(N',T')$ was derived from $D(N,T)$ by deleting a subset of arcs and vertices, it is clear that $D(N',T')$ remains a display graph.

ϕ' is an embedding function. As ϕ' is the restriction of ϕ to T', and no arc in a path $\phi(uv)$ was deleted unless the arc uv was deleted as well (and similarly for vertices $\phi(u)$), ϕ' is still a function that maps vertices of T' to vertices of N' and arcs of T' to directed paths of N'. The other properties of an embedding function (such as all paths being arc-disjoint) follow immediately from the fact that these properties hold for ϕ. Thus, ϕ' is an embedding function on $D(N',T')$.

ι' is an $(S', \mathcal{Y})$-isolabelling. Note that the properties of an isolabelling follow immediately from construction of ι', the fact that $g(v) = v$ for all $v \in S'$ and the fact that ι is an $(S, \mathcal{Y})$-isolabelling. To see

that the image of ι' in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ contains S', we prove that, for every $z \in S'$, there is some $u \in V(D(N', T'))$ for which $\iota'(u) = z$. To see this, consider the vertex $u \in V(D(N, T))$ for which $\iota(u) = z$. Since $z \in S'$ we then have $g(\iota(u)) = g(z) = z \notin \mathcal{Y}$ and, thus, u cannot become redundant in the construction of χ'. Hence, u is also a vertex of $D(N', T')$, and $\iota'(u) = g(\iota(u)) = z$ as required.

Same degrees in $D(N', T')$ as in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Let $u \in V(D(N, T))$ with $\iota(u) \in S'$. Observe that no arc incident to u is deleted when constructing $D(N', T')$ and, thus, u has the same in-and out-degrees in $D(N', T')$ as it does in $D(N, T)$, that is, the same in- and out-degrees as $\iota(u) = \iota'(u)$.

Out-arcs in $D(N', T')$. Let $u \in V(T')$ with $\iota'(u) \neq \iota'(\phi'(u))$. In particular, there is no $y \in \mathcal{Y}$ with $\iota'(u) = \iota'(\phi'(u)) = y$. Then, no arc incident to u was deleted (as for any such arc a the path $\phi(a)$ contains $\phi(u)$ by Definition 2(a)) and, so, u has the same out-degree in $D(N', T')$ as in $D(N, T)$. Moreover, u has out-degree 2 in $D(N, T)$ since $\iota(u) \neq \iota(\phi(u))$ (otherwise $\iota'(u) = \iota'(\phi'(u))$ by construction of ϕ' and ι') and, thus, in $D(N', T')$. □

Lemma 4 (transitivity of restrictions)**.** *Let χ be an $(S, \mathcal{Y})$-containment structure for some S and $\mathcal{Y}$, let $S'' \subseteq S' \subseteq S$ and let $g' : S \cup \mathcal{Y} \to S' \cup \mathcal{Y}$ and $g'' : S' \cup \mathcal{Y} \to S'' \cup \mathcal{Y}$ be restriction functions. Let χ' be the g'-restriction of χ and let χ'' be the g''-restriction of χ'. Then, χ'' is the $(g'' \circ g')$-restriction of χ.*

Proof: First, we show that $g := g'' \circ g'$ is a restriction function. To this end, let $v \in S \cup \mathcal{Y}$. If $v \in S''$, then $v \in S' \cap S$ and, thus, $g(v) = g''(g'(v)) = g''(v) = v$ since both g' and g'' are restriction functions. Otherwise, $v \notin S''$ and either $v \in S'$, in which case $g'(v) = v \notin S''$, or $g'(v) \in \mathcal{Y}$. In either case $g'(v) \notin S''$ and so $g(v) = g''(g'(v)) \in \mathcal{Y}$.

In the following, let $\chi' = (D(N', T'), \phi', \iota')$, let $\chi'' = (D(N'', T''), \phi'', \iota'')$ and let $\iota_{g'}$ be as described in Definition 7, that is, $\iota_{g'}(u) = g'(\iota(u))$ for all $u \in V(D(N, T))$ (and analogously for $\iota'_{g''}$ and ι_g). Then, by Definition 7, $\iota' = \iota_{g'}$ and $\iota'' = \iota'_{g''}$. Note that, for any $u \in V(D(N', T'))$,

$$\iota'_{g''}(u) = g''(\iota'(u)) = g''(\iota_{g'}(u)) = g''(g'(\iota(u))) = g(\iota(u)) = \iota_g(u). \tag{1}$$

Next, we show that χ'' equals the g-restriction $\chi^* = (D(N^*, T^*), \phi^*, \iota^*)$ of χ. To this end, we first prove that $D(N^*, T^*) = D(N'', T'')$. As both display graphs are subgraphs of $D(N, T)$, it is enough to show that any arc or vertex is deleted in the construction of $D(N^*, T^*)$ from $D(N, T)$ if and only if it is deleted in the construction of $D(N', T')$ from $D(N, T)$ or of $D(N'', T'')$ from $D(N', T')$.

Arcs: Let uv be an arc of $D(N, T)$. If $uv \in A(T)$, then let $Q_{uv} := \{u, v\} \cup V(\phi(uv))$ (that is, Q_{uv} contains u, v and all vertices in the path $\phi(uv)$). If uv is an arc of N, then let Q_{uv} be the set containing u and v together with any u' and v' for which uv is in the path $\phi(u'v')$ (by Definition 2, there is at most one such arc $u'v'$). By Observation 1, uv is deleted in the construction of $D(N^*, T^*)$ if and only if there is some $y \in Y$ with $\iota_g(Q_{uv}) = \{y\}$.

First, assume that uv is in $D(N'', T'')$ but not in $D(N^*, T^*)$. Since $D(N'', T'')$ is a subgraph of $D(N', T')$, we know that uv is also in $D(N', T')$. If $uv \in A(T)$, then all vertices of $\phi'(uv) = \phi(uv)$ are still in $D(N', T')$ as ϕ' is an embedding function on $D(N', T')$. If $uv \in A(N)$ and $Q_{uv} = \{u, v, u', v'\}$ and any of u' and v' is not in $D(N', T')$, then the arc $u'v' \in A(T)$ was deleted in the construction of $D(N', T')$ and, by Observation 2, so was uv, contradicting uv being in $D(N', T')$. Thus, all vertices of Q_{uv} are in $D(N', T')$ and, by (1), $\iota'_{g''}(Q_{uv}) = \iota_g(Q_{uv}) = \{y\}$ for some $y \in \mathcal{Y}$. Then uv is deleted in the construction of $D(N'', T'')$.

Second, assume that uv is in $D(N^*,T^*)$ but not in $D(N'',T'')$. If uv is in $D(N',T')$ then uv is redundant with respect to $(D(N',T'),\phi',\iota')$, that is, $Q_{uv} \subseteq V(D(N',T'))$ and, by Observation 1, there is some $y \in \mathcal{Y}$ with $\{y\} = \iota'_{g''}(Q_{uv}) = \iota_g(Q_{uv})$ (using (1)). But then, uv is also deleted in the construction of $D(N^*,T^*)$. If uv is not in $D(N',T')$, then $\iota_{g'}(Q_{uv}) = \{y\}$ for some $y \in \mathcal{Y}$, implying $\iota_g(Q_{uv}) = g(\iota(Q_{uv})) = g''(g'(\iota(Q_{uv})) = g''(\iota_{g'}(Q_{uv})) = g''(\{y\}) = \{g''(y)\}$. As $y \in \mathcal{Y}$ implies $g''(y) \in \mathcal{Y}$, we know that uv is deleted in the construction of $D(N^*,T^*)$.

Vertices: Since each arc of $D(N,T)$ is in $D(N^*,T^*)$ if and only if it is in $D(N'',T'')$, all vertices with at least one incident arc in $D(N^*,T^*)$ are in $D(N'',T'')$ and vice versa. It remains to consider the isolated vertices. To this end, let $v \in V(D(N,T))$ with no incident arcs in $D(N^*,T^*)$ and let $Q_v := \{v,\phi(v)\}$ if $v \in V(T)$ and let Q_v be the set containing v and any u such that $v = \phi(u)$ if $v \in V(N)$. By definition of redundant vertices, either both vertices of Q_v are deleted in the construction of $D(N^*,T^*)$ or neither is. As $D(N^*,T^*)$ and $D(N'',T'')$ have the same arcs, we may assume neither element of Q_v has any incident arcs. Then, v is deleted in the construction of $D(N^*,T^*)$ if and only if $\iota_g(Q_v) = \{y\}$ for some $y \in \mathcal{Y}$. Now, if Q_v intersects $V(D(N',T')$, then $Q_v \subseteq V(D(N',T'))$. Then, by (1), $\iota'_{g''}(Q_v) = \iota_g(Q_v)$ implying that v is in $D(N'',T'')$ if and only if it is in $D(N^*,T^*)$. Otherwise, $Q_v \cap V(D(N',T')) = \varnothing$ (in particular, $v \notin V(D(N',T'))$), implying $\iota_{g'}(Q_v) = \{y\}$ for some $y \in \mathcal{Y}$. As in the Arc-case, $\iota_g(Q_v) = \{g''(y)\} = \{y'\}$ for some $y' \in \mathcal{Y}$. Thus, v is neither in $D(N^*,T^*)$ nor in $D(N'',T'')$.

Since ϕ^* is the restriction of ϕ to T^* and ϕ'' is the restriction of ϕ to T'' (via ϕ'), we also have $\phi^* = \phi''$. By (1), $\iota'_{g''}(u) = \iota_g(u)$ for any vertex u in $D(N',T')$ and, thus, $\iota''(u) = \iota^*(u)$ for any vertex u in $D(N'',T'')$ (as ι'' and ι^* are just restrictions of $\iota'_{g''}$ and ι_g respectively). Therefore, $\iota^* = \iota''$ and we have proved that $\chi^* = \chi''$, as required. □

3.4 Well-behaved containment structures

At this point, we note some additional properties that it will be helpful to assume for $(S,\mathcal{Y})$-containment structures. These properties are summarized in the concept of "well-behavedness" and, in what follows, we will restrict our attention to such containment structures. While not directly implied by the definition of a containment structure, the properties effectively ensure that containment structures behave "as expected". In particular, our dynamic programming algorithm will work with well-behaved signatures.

Definition 8. *An $(S,\mathcal{Y})$-containment structure $(D(N,T),\phi,\iota)$ is called* well-behaved *if*
(a) $D(N,T)$ contains no redundant arcs or vertices;
(b) For all arcs uv in $D(N,T)$ with $\iota(u),\iota(v) \in \mathcal{Y}$, we have $\iota(u) = \iota(v)$;
(c) For all $u,v \in V(D(N,T))$ such that $\iota(u),\iota(v) \in S$ and $D(N,T)$ has a u-v-path, $D_{\text{IN}}(N_{\text{IN}},T_{\text{IN}})$ has an $\iota(u)$-$\iota(v)$-path.

Here we prove a number of properties of well-behaved containment structures, that allow us to focus on them going forward.

Lemma 5. *Let $\chi := (D(N,T),\phi,\iota)$ be a $(V(D_{\text{IN}}(N_{\text{IN}},T_{\text{IN}})),\mathcal{Y})$-containment structure with $\iota^{-1}(\mathcal{Y}) = \varnothing$. Then, χ is well-behaved.*

Proof: Since $\iota^{-1}(\mathcal{Y}) = \varnothing$ and $S = V(D_{\text{IN}}(N_{\text{IN}},T_{\text{IN}}))$, we know that ι is an isomorphism between $D(N,T)$ and $V(D_{\text{IN}}(N_{\text{IN}},T_{\text{IN}}))$, implying Definition 8(c), while (a) and (b) are direct consequences of $\iota^{-1}(\mathcal{Y}) = \varnothing$. □

Lemma 6. *Let $\chi = (D(N,T), \phi, \iota)$ be a well-behaved $(S, \mathcal{Y})$-containment structure and let g be a restriction function such that, for all arcs uv of $D(N,T)$ with $g(\iota(u)), g(\iota(v)) \in \mathcal{Y}$, we have $g(\iota(u)) = g(\iota(v))$. Then, the g-restriction σ of χ is also well-behaved.*

Essentially, this condition says that relabelling according to g does not immediately create any arcs uv for which u and v are labelled with different elements of $\mathcal{Y}$.

Proof: We will show that $\sigma = (D(N',T'), \phi', \iota' = \iota \circ g)$ satisfies Definition 8(a)-(c).
(a) follows by Definition 7, as redundant arcs and vertices are deleted when constructing σ.
(b)**:** Let uv be an arc of $D(N',T')$ with $\iota'(u), \iota'(v) \in \mathcal{Y}$. Then, by the condition of the lemma, $\iota'(u) = \iota'(v)$.
(c)**:** Let $u, v \in V(D(N',T'))$ such that $\iota'(u), \iota'(v) \notin \mathcal{Y}$ (that is, $\iota'(u), \iota'(v) \in S'$ for $S' := \mathrm{img}(g) \setminus \mathcal{Y}$) and there is a u-v-path p in $D(N',T')$. Then, by Definition 7, $\iota(u) = g(\iota(u)) = \iota'(u)$ and, similarly, $\iota(v) = \iota'(v)$. Moreover, as $D(N',T')$ is a subgraph of $D(N,T)$, the latter also contains p. Finally, since χ is well-behaved, there is a path in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ from $\iota(u) = \iota'(u)$ to $\iota(v) = \iota'(v)$. □

3.5 Partial solution and valid signatures

As with any dynamic programming algorithm, we need some way to decide which signatures are "correct" before we have actually found a solution. As such, we need a notion of a "partial solution". Much as we may think of a signature for a bag (P, S, F) as corresponding to the $(P \to \text{PAST}, F \to \text{FUTURE})$-restriction of some solution, we may think of a partial solution as the $(F \to \text{FUTURE})$-restriction of some solution. That is, a partial solution is a $(P \cup S, \{\text{FUTURE}\})$-containment structure that roughly corresponds to what would happen if we took a solution and "forgot" some of the details about the vertices in F. A partial solution is then a "witness" for a given signature for (P, S, F) if that signature can in turn be derived from the partial solution by "forgetting" details about the vertices in P. In this case we call the signature "valid". This is defined precisely below.

Definition 9 (partial solution, signature, valid)**.** *Let (P, S, F) be a bag in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Then any $(P \cup S, \{\text{FUTURE}\})$-containment structure is called* F-partial solution *(or simply* partial solution*) for (P, S, F). A well-behaved signature σ for (P, S, F) is called* valid *if σ is the $(P \to \text{PAST})$-restriction of a well-behaved partial solution ψ for (P, S, F). We call ψ a* witness *for σ.*

Lemma 7. *Let $(N_{\text{IN}}, T_{\text{IN}})$ be a* YES*-instance of* TREE CONTAINMENT *and let (P, S, F) be a bag in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Then, there is a well-behaved F-partial solution ψ, and a valid signature σ for (P, S, F).*

Proof: By Lemma 5, there is a well-behaved $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \mathcal{Y})$-containment structure ψ^* with $\iota^{-1}(\mathcal{Y}) = \varnothing$. Clearly, ψ^* is also a $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing)$-containment structure. Now, let ψ be the $(F \to \text{FUTURE})$-restriction of ψ^* and note that ψ is an F-partial solution. Clearly, $x, y \in \{\text{FUTURE}\} \Rightarrow x = y$ and, so, Lemma 6 applies, showing that ψ is well-behaved. Finally, let σ be the $(P \to \text{PAST})$-restriction of ψ which is valid since ψ is well-behaved. □

Lemma 8. *$(N_{\text{IN}}, T_{\text{IN}})$ is a* YES*-instance of* TREE CONTAINMENT *if and only if there is a valid signature $\sigma := (D(N,T), \phi, \iota)$ for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\iota^{-1}(\text{FUTURE}) = \varnothing$.*

Proof: By Lemma 2 and Lemma 5, $(N_{\text{IN}}, T_{\text{IN}})$ is a YES-instance if and only if there is a well-behaved $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \mathcal{Y})$-containment structure $\psi = (D(N,T), \phi, \iota)$ with $\iota^{-1}(\{\mathcal{Y}\}) = \varnothing$, or equivalently a

well-behaved $\varnothing$-partial solution $\psi = (D(N,T), \phi, \iota)$ with $\iota^{-1}(\{\text{FUTURE}\}) = \varnothing$. First, suppose such a partial solution exists and let $\sigma := ((D(N',T'), \phi', \iota')$ be the $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \to \text{PAST})$-restriction of ψ. Then, σ is a valid signature for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ and, by construction, ${\iota'}^{-1}(\text{FUTURE}) = \varnothing$. (In fact $D(N',T')$ is the empty graph since all arcs and vertices of $D(N,T)$ become PAST-redundant; but we do not use that fact here).

For the converse, consider a valid signature $\sigma := (D(N,T), \phi, \iota)$ for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\iota^{-1}(\text{FUTURE}) = \varnothing$. By Definition 9, σ is the $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \to \text{PAST})$-restriction of a well-behaved $\varnothing$-partial solution $\psi := ((D(N',T'), \phi', \iota')$. It remains to show that ${\iota'}^{-1}(\text{FUTURE}) = \varnothing$. Towards a contradiction, assume that $D(N',T')$ has a vertex u with $\iota'(u) = \text{FUTURE}$. Since $u \notin \iota^{-1}(\text{FUTURE}) = \varnothing$, u is FUTURE-redundant after applying $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \to \text{PAST}$. However, applying $V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \to \text{PAST}$ cannot make a vertex FUTURE-redundant that was not previously FUTURE-redundant (as no new vertex gains the label FUTURE). Thus, u is FUTURE-redundant in ψ, contradicting Definition 8(a). □

Lemma 6, Lemma 7 and Lemma 8 show that an instance $(N_{\text{IN}}, T_{\text{IN}})$ of TREE CONTAINMENT is a YES-instance if and only if there is a well-behaved valid signature $\sigma := (D(N,T), \phi, \iota)$ for the root bag $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\iota^{-1}(\text{FUTURE}) = \varnothing$. Thus in order to solve an instance of TREE CONTAINMENT, it is enough to decide for each bag (P,S,F) in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, and for each well-behaved signature σ for (P,S,F), whether σ is valid.

3.6 Determining valid signatures

With the formal definitions of valid signatures and restrictions in place, we can now show how to determine whether a well-behaved signature for a bag (P,S,F) is valid, assuming we know this for all signatures on the child bag(s). As is common in dynamic programming techniques, we take advantage of the structure of a nice tree decomposition. The following lemmas describe the exact conditions for Leaf, Forget and Introduce bags while the additional terminology required for Join bags is deferred to Section 3.7.

Lemma 9. *Let (P,S,F) correspond to a Leaf bag in the tree decomposition i.e. $P = S = \varnothing$, $F = V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and (P,S,F) has no children. Let $\sigma := (D(N,T), \phi, \iota)$ be a well-behaved signature for (P,S,F). Then, σ is valid if and only if $\iota^{-1}(\text{PAST}) = \varnothing$.*

Proof: Suppose first that σ is valid. By Definition 9, σ is the $(P \to \text{PAST})$-restriction of a well-behaved F-partial solution (that is, $(P \cup S, \{\text{FUTURE}\})$-containment structure) $\psi := D(N',T'), \phi', \iota')$. Then $\iota'(u) \neq \text{PAST}$ for every vertex u in $D(N',T')$ and, as $P = \varnothing$, this remains true after applying $P \to \text{PAST}$. It follows that $\iota(u) \neq \text{PAST}$ for every vertex $u \in D(N,T)$, as required.

Conversely, suppose $\iota^{-1}(\text{PAST}) = \varnothing$. Since $P = \varnothing$, the $(S, \{\text{PAST}, \text{FUTURE}\})$-containment structure σ is also a $(P \cup S, \{\text{FUTURE}\})$-containment structure, that is, σ is an F-partial solution. To show that σ is valid, we prove that σ is the $(P \to \text{PAST})$-restriction of itself, that is, σ is a witness for σ. To this end, observe that applying $P \to \text{PAST}$ does not change the isolabelling and, by Definition 8, σ contains no redundant arcs or vertices. Thus, applying $P \to \text{PAST}$ and removing redundant arcs and vertices does not change σ. □

Lemma 10. *Let (P,S,F) correspond to a Forget bag in the tree decomposition with child bag (P',S',F), i.e. $P = P' \cup \{z\}$ and $S = S' \setminus \{z\}$ for some $z \in S'$. Let σ be a well-behaved signature for (P,S,F). Then, σ is valid if and only if σ is the $(z \to \text{PAST})$-restriction of a valid signature σ' for (P',S',F).*

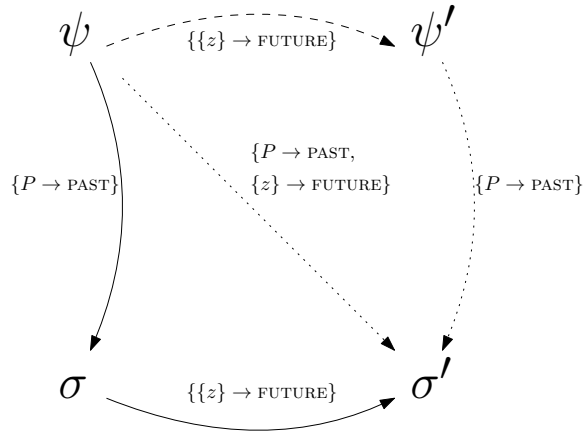

Figure 5: Illustration of proof for Introduce bags, validity of σ implies validity of σ'. Solid lines are restriction relations that we may assume; the dashed line shows the construction of ψ' from ψ; dotted lines are relations we can infer using transitivity.

Proof: Suppose first that σ is the $(z \to \text{PAST})$-restriction of some valid signature σ' for (P', S', F). By Definition 9, σ' has a witness ψ' (that is, ψ' is a well-behaved F-partial solution for (P', S', F) whose $P' \to \text{PAST}$-restriction is σ'). By Lemma 4, σ is the $(P \to \text{PAST})$-restriction of ψ', and so σ is valid.

For the converse, suppose that σ is valid and let ψ be a witness of σ (that is, ψ is a well-behaved F-partial solution for (P, S, F) and σ is the $(P \to \text{PAST})$-restriction of ψ). Then, the $(P' \to \text{PAST})$-restriction σ' of ψ is a valid signature for (P', S', F) and, by Lemma 4, the $(z \to \text{PAST})$-restriction of σ' is the $(P \to \text{PAST})$-restriction of ψ, that is, σ. □

Lemma 11. *Let (P, S, F) correspond to an Introduce bag in the tree decomposition with child bag (P, S', F'), i.e. $S' = S \setminus \{z\}$ and $F' = F \cup \{z\}$ for some $z \in S$. Let σ be a well-behaved signature for (P, S, F). Then, σ is valid if and only if the $(z \to \text{FUTURE})$-restriction σ' of σ is a valid signature for (P, S', F').*

Proof: Suppose first that σ is a valid signature and let ψ be a witness for σ (that is, ψ is a well-behaved F-partial solution for which σ is the $(P \to \text{PAST})$-restriction). Let ψ' be the $(z \to \text{FUTURE})$-restriction of ψ which, by Lemma 6, is a well-behaved F'-partial solution. Let σ^* denote the $(P \to \text{PAST})$-restriction of ψ' and note that σ^* is a valid signature for (P, S', F'). By Lemma 4, σ^* is also the $(P \to \text{PAST}, \{z\} \to \text{FUTURE})$-restriction of ψ and the $(z \to \text{FUTURE})$-restriction σ' of σ is also the $(P \to \text{PAST}, \{z\} \to \text{FUTURE})$-restriction of ψ. Thus, $\sigma' = \sigma^*$ and so σ' is valid, as required (see Fig. 5).

For the converse, let $\sigma =: (D(N, T), \phi, \iota)$ and let $\sigma' =: (D(N', T'), \phi', \iota')$ be a valid signature for (P, S', F'). By Definition 9, σ' is the $(P \to \text{PAST})$-restriction of some well-behaved F'-partial solution $\psi' := (D(N'_0, T'_0), \phi'_0, \iota'_0)$. In the following, σ, σ' and ψ' will guide us in constructing an F-partial solution ψ of which ψ' is the $(z \to \text{FUTURE})$-restriction and, more importantly, of which σ is the $(P \to \text{PAST})$-restriction. Then, this implies that σ is valid.

To begin the construction, let V_z and A_z denote the the set of vertices and arcs, respectively, that are in $D(N, T)$ but not in $D(N', T')$ (that is, the vertices and arcs that are deleted when deriving σ' from σ). Letting g be the function $z \to \text{FUTURE}$, we may assume that $g(\iota(v)) = \text{FUTURE}$ for all vertices v in V_z

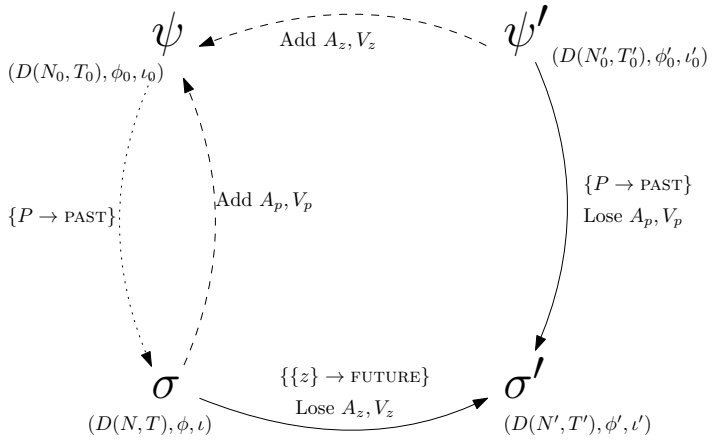

Figure 6: Illustration of proof for Introduce bags, validity of σ' implies validity of σ. Solid lines are restriction relations that we may assume; the dashed lines show the construction of ψ from ψ' or σ (we do not describe the construction of ϕ_0 or ι_0 in the figure, only the construction of $D(N_0, T_0)$ by adding arcs and vertices); the dotted line shows the relation we want to prove, that σ is the $\{P \to \text{PAST}\}$-restriction of ψ.

or $V(A_z)$ and $g(\iota(V(\phi(uv)))) = \{\text{FUTURE}\}$ for any tree arc uv in A_z as well. Similarly, let V_p and A_p be the vertices and arcs, respectively, that are deleted from $D(N_0', T_0')$ in the construction of $D(N', T')$. Letting h be the function $P \to \text{PAST}$, we may assume that $h(\iota(v)) = \text{PAST}$ for all vertices v in V_p or $V(A_p)$, and $h(\iota(V(\phi(uv)))) = \{\text{PAST}\}$ for any tree arc uv in A_p as well.

We now construct the F-partial solution $\psi := (D(N_0, T_0), \phi_0, \iota_0)$ as follows. First, let $D(N_0, T_0)$ be the graph derived from $D(N_0', T_0')$ by adding all vertices and arcs of V_z and A_z. Equivalently, we may say we construct $D(N_0, T_0)$ by adding the arcs and vertices of V_p and A_p to $D(N, T)$, or by adding the arcs and vertices of V_z, V_p, A_z, A_p to $D(N', T')$. Let ϕ_0 be defined as the 'union' of ϕ and ϕ_0' - that is, $\phi_0(uv) = \phi(uv)$ if uv is an arc in T, and $\phi_0(uv) = \phi_0'(uv)$ if uv is an arc in T_0' (if uv is an arc of both T and T_0', then these are the same, as uv is in T' and $\phi(uv) = \phi'(uv) = \phi_0'(uv)$). Similarly $\phi_0(u) = \phi(u)$ if u is a vertex in T, and $\phi_0(u) = \phi_0'(u)$ if u is a vertex in T_0'. Finally let $\iota_0 : V(D(N_0, T_0)) \to P \cup S \cup \{\text{FUTURE}\}$ be defined as follows: $\iota_0(v) = \iota(v)$ if $v \in V_z$ or $\iota'(v) = \text{FUTURE}$, $\iota_0(v) = \iota_0'(v)$ if $v \in V_p$ or $\iota'(v) = \text{PAST}$, and $\iota_0(v) = \iota_0'(v) = \iota'(v) = \iota(v)$ otherwise. (Note that $\iota_0(v) = \text{FUTURE}$ is possible for some vertices, if $\iota(v) = \text{FUTURE}$, but $\iota_0(v) = \text{PAST}$ is not possible as $\iota_0'(v) \neq \text{PAST}$ for any v.)

We note here that for any vertex v in $D(N_0, T_0)$ with $\iota_0(v) \in S \cup \{\text{FUTURE}\}$, v is a vertex of $D(N, T)$ and $\iota_0(v) = \iota(v)$. Indeed, v cannot be in V_p, nor can it hold that $\iota'(v) = \text{PAST}$ if v is in $D(N', T')$, as both cases would imply $\iota_0(v) = \iota_0'(v) \in P$. By construction, in all other cases $\iota_0(v) = \iota(v)$. Furthermore, for any arc a incident to v in $D(N_0, T_0)$, a is an arc in $D(N, T)$ (since $a \in A_p$ would imply $\iota_0'(v) \in P$ and hence either $v \in V_p$ or $\iota'(v) = \text{PAST}$), and thus $\phi_0(a) = \phi(a)$. A similar argument shows that for any vertex v in $D(N_0, T_0)$ with $\iota(u) \in P$, v is a vertex of $D(N_0', T_0')$, $\iota_0(v) = \iota_0'(v)$, and $\phi_0(a) = \phi_0'(a)$ for any arc a in $D(N_0, T_0)$ incident to v.

We will now show that ψ is indeed a well-behaved F-partial solution; afterwards we will show that σ is the $\{P \to \text{PAST}\}$-restriction of ψ, from which it follows that σ is valid. (A similar argument can be used

to show that ψ' is the $\{\{z\} \to \text{FUTURE}\}$-restriction of ψ, but we will not need this fact so we do not prove it here.)

Claim 1. *ψ is a well-behaved F-partial solution.*

Proof: We first show that ψ satisfies one of the properties of a well-behaved $(P \cup S, \{\text{FUTURE}\})$-containment structure

- **for any $u, v \in V(D(N_0, T_0))$ with $\iota_0(u), \iota_0(v) \in P \cup S$, if $D(N_0, T_0)$ has a path from u to v then there is a path from $\iota_0(u)$ to $\iota_0(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$:** Suppose for a contradiction that this is not the case, and let $u, v \in V(D(N_0, T_0))$ be such that $\iota_0(u), \iota_0(v) \in P \cup S$ and there is a path from u to v in $D(N_0, T_0))$, but no path from $\iota_0(u)$ to $\iota_0(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, choosing uv such that the length of the $u - v$ path is minimal. If uv is an arc in $D(N_0, T_0)$, then uv is an arc in $D(N, T)$ or $D(N'_0, T'_0)$, and the existence of a path from $\iota(u)$ to $\iota(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ follows from the fact that σ or σ'_0 is well-behaved. Otherwise, by choice of u, v we may assume all internal vertices u' on the path from u to v satisfy $\iota_0(u') = \text{FUTURE}$. Then every arc on this path is an arc in $D(N, T)$ and so again, we have a path from $\iota_0(u) = \iota(u)$ to $\iota_0(v) = \iota(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ by the fact that σ is well-behaved.

- **$D(N_0, T_0)$ is a display graph:**

  - **$D(N_0, T_0)$ is acyclic:** Suppose for a contradiction that $D(N_0, T_0)$ contains a directed cycle. If $\iota_0(u) \in P \cup S$ for some vertex u in this cycle, then as shown above the existence of a path from u to u in $D(N_0, T_0)$ implies the existence of a path from $\iota_0(u)$ to $\iota_0(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, contradicting the fact that $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ is acyclic.

    It remains to consider the case that every vertex z in this cycle has $\iota_0(z) = \text{FUTURE}$. But in this case every arc of the cycle is also an arc in $D(N, T)$, and so $D(N, T)$ contains a cycle, a contradiction as $D(N, T)$ is a display graph.

  - **T_0 is an out-forest:** We first observe that for every vertex u in $D(N_0, T_0)$, either all its incident arcs in $D(N_0, T_0)$ are also arcs of $D(N, T)$, or they are all arcs of $D(N'_0, T'_0)$. Indeed, suppose for a contradiction that this is not the case, then there exist arcs $a \in A_p$, $a' \in A_z$ that share a vertex u. But by construction, $a \in A_p$ implies that $u \in V_p$ or $\iota'(u) = \text{PAST}$ and so $\iota_0(u) \in P$, while $a' \in A_z$ implies $\iota_0(u) \in S \cup \{\text{FUTURE}\}$, a contradiction.

    Now it remains to observe that every vertex in T_0 has in-degree at most 1, as either all its incident arcs are in T or they are all in T'_0. This, together with the fact that $D(N_0, T_0)$ is acyclic, implies that T_0 is an out-forest.

  - **Every vertex in $D(N_0, T_0)$ has in- and out-degree at most 2 and total degree at most 3:** As argued above, every vertex in $D(N_0, T_0)$ has all its incident arcs in $D(N, T)$ or in $D(N'_0, T'_0)$, and so this property follows from the fact that it holds for $D(N, T)$ and $D(N'_0, T'_0)$

  - **Any vertex in $V(N_0) \cap V(T_0)$ has out-degree 0 and in-degree at most 1 in each of T_0 and N_0:** Again, this follows from the fact that all incident arcs of a vertex in $D(N_0, T_0)$ belong to one of $D(N, T)$, $D(N'_0, T'_0)$.

- **ϕ_0 is an embedding function on $D(N_0, T_0)$:**

- **For each** $u \in V(T_0)$**,** $\phi_0(u) \in V(N_0)$ **and, for each arc** $uv \in A(T_0)$**,** $\phi_0(uv)$ **is a directed** $\phi_0(u)$**-**$\phi_0(v)$**-path in** N_0**:** This follows immediately from the construction of ϕ_0 and the fact that ϕ and ϕ'_0 are embedding functions.
- **for any distinct** $u, v \in V(T_0)$**,** $\phi_0(u) \neq \phi_0(v)$: Note that if a tree vertex u is in V_p then, by definition of PAST-redundant, so is $\phi'_0(u)$. Similarly if $u \in V_z$ then $\phi(u) \in V_z$. So now for two vertices $u, v \in V(T_0)$, if $u \in V_p$ and $v \in V_z$ then $\phi_0(u) = \phi'_0(u) \in V_p$ and $\phi_0(v) = \phi(v) \in V_z$, and so $\phi_0(u) \neq \phi_0(v)$. Otherwise, u, v are either both in $D(N, T)$ or both in $D(N'_0, T'_0)$, and $\phi_0(u) \neq \phi_0(v)$ follows from the fact that ϕ and ϕ'_0 are embedding functions.
- **for any** $u \in V(T_0) \cap V(N_0)$**,** $\phi_0(u) = u$**:** Follows immediately from the construction of ϕ_0.
- **the paths** $\{\phi_0(uv) \mid uv \in A(T_0)\}$ **are arc-disjoint:** Observe that by construction that if a tree arc uv is in A_p, then so is every arc in $\phi_0(uv) = \phi'_0(uv)$. Similarly if uv is in A_z then so is every arc in $\phi_0(uv) = \phi(uv)$. So consider two distinct tree arcs $uv, u'v' \in A(T_0)$. If $uv \in A_p, u'v' \in A_z$ then the arcs of the paths $\phi_0(uv), \phi_0(u'v')$ are in A_p, A_z respectively, so $\phi_0(uv), \phi_0(u'v')$ are arc-disjoint. Otherwise, we may assume both uv and $u'v'$ are arcs in one of $D(N, T), D(N'_0, T'_0)$, from which the claim follows by the fact that ϕ and ϕ'_0 are embedding functions.
- **for any distinct** $p, q \in A(T_0)$**,** $\phi_0(p)$ **and** $\phi_0(q)$ **share a vertex** z' **only if** p **and** q **share a vertex** w **with** $z' = \phi_0(w)$**:** Suppose $\phi_0(p)$ and $\phi_0(q)$ share a vertex z'. As argued previously, all incident arcs to z' must be in one of $D(N, T), D(N'_0, T'_0)$. Thus in particular, z' cannot have incident arcs from both A_p and A_z. Then since $\phi_0(p)$ and $\phi_0(q)$ both contain arcs incident to z', we must have that $p, q \notin A_p$ or $p, q \notin A_z$. Then either $\phi_0(p) = \phi(p)$ and $\phi_0(q) = \phi(q)$, or $\phi_0(p) = \phi'_0(p)$ and $\phi'_0(q)$. Then the claim follows from the fact that ϕ and ϕ'_0 are embedding functions.

- ι_0 **is a** $(P \cup S, \{\textsc{future}\})$**-isolabelling:**
  - **For** $u \in V(D(N_0, T_0))$ **with** $\iota_0(u) \neq \textsc{future}$**,** $\iota_0(u) \in V(N_{\text{IN}})$ **only if** $u \in V(N_0)$ **and** $\iota_0(u) \in V(T_{\text{IN}})$ **only if** $u \in V(T_0)$ **:** Follows immediately from the construction of ι_0.
  - **For** $u, v \in V(D(N_0, T_0))$ **with** $\iota_0(u), \iota_0(v) \neq \textsc{future}$**,** $\iota_0(u) = \iota_0(v)$ **only if** $u = v$**:** Suppose $u \neq v$ and $\iota_0(u), \iota_0(v) \neq$ FUTURE; we will show $\iota_0(u) \neq \iota_0(v)$. If $u \in V_p$ and $v \in V_z$, then by construction $\iota_0(u) = \iota(u) \in \{z, \text{FUTURE}\}$ (and so in fact $\iota_0(u) = z$) and $\iota_0(v) = \iota'_0(v) \in P$. Thus $\iota_0(u) \neq \iota_0(v)$. Otherwise, we may assume u and v are both vertices in $D(N, T)$ or in $D(N'_0, T'_0)$. Then $\iota_0(u) \neq \iota_0(v)$ follows from the fact that ι and ι'_0 are isolabelings.
  - **For** $u, v \in V(D(N_0, T_0))$ **with** $\iota_0(u), \iota_0(v) \neq \textsc{future}$**, the arc** uv **is in** $D(N_0, T_0)$ **if and only if** $\iota_0(u)\iota_0(v)$ **is in** $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ **:** Follows immediately from the construction of ι_0 and the fact that ι, ι'_0 are isolabelings.
  - ι_0 **is surjective onto** $P \cup S$**:** Consider any $w \in P \cup S$. If $w \in P$, there is $u \in V(D(N'_0, T'_0))$ with $\iota'_0(u) = w$. For such u we have either $u \in V_p$ or $\iota'(u) =$ PAST. In either case we have $\iota_0(u) = \iota'_0(u) = w$. Similarly if $w \in S$, there is $u \in V(D(N, T))$ with $\iota(u) = w$ and either $\iota'(u) = \iota(u)$ (if $w \in S'$) or $u \in V_z$ or $\iota'(u) =$ FUTURE (if $w = z$), and so $\iota_0(u) = \iota(u) = w$.

- **Each vertex u with $\iota_0(u) \in P \cup S$ has the same in- and out-degree in $D(N_0, T_0)$ as $\iota_0(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$:** As previously shown, all incident arcs of u in $D(N_0, T_0)$ belong to at least one of $D(N, T), D(N'_0, T'_0)$. Moreover if $\iota_0(u) = \iota(u)$ then all incident arcs are in $D(N, T)$ and if $\iota_0(u) = \iota'_0(u)$ then all incident arcs are in $D(N'_0, T'_0)$. Then this property follows from the fact that σ and σ'_0 are containment structures.

- **Each vertex u of T_0 with $\iota_0(u) \neq \iota(\phi_0(u))$ has 2 out-arcs in $D(N_0, T_0)$:** In the case that $\iota_0(u) \in P \cup S$, this follows from previously-shown properties. We may assume u is an internal vertex of T_0 (as otherwise $u \in V(T_0) \cap V(N_0)$ and $\phi_0(u) = u$). Then $\iota_0(u)$ is also an internal vertex of T_{IN}, and u has the same in-and out-degree as $\iota_0(u)$. Thus in particular u has out-degree 2, as T_0 is binary.

  For the case that $\iota_0(u) = \text{FUTURE}$, by construction $\iota_0(u) = \iota(u)$. Furthermore $\iota(\phi(u)) \neq \text{FUTURE}$ (as this would imply $\iota_0(\phi_0(u)) = \iota(\phi(u) = \text{FUTURE} = \iota_0(u)$, a contradiction). Then as σ is a $(S, \{\text{PAST}, \text{FUTURE}\})$-containment structure, u has out-degree 2 in $D(N, T)$, and therefore in $D(N_0, T_0)$.

  The above conditions show that ψ is a $(P \cup S, \{\text{FUTURE}\})$-containment structure, i.e. an F-partial solution. Next we show that ψ is well-behaved:

- **$D(N_0, T_0)$ contains no redundant arcs or vertices:** Suppose for a contradiction that $D(N_0, T_0)$ contains a FUTURE-redundant arc or vertex a. We will show that such an arc or vertex is also redundant w.r.t σ, a contradiction as σ is well-behaved. Consider the case that a is an arc uv Then $\iota_0(u) = \iota_0(v) = \text{FUTURE}$. It follows by construction that u, v are vertices of $D(N, T)$ and $\iota(u) = \iota(v) = \text{FUTURE}$. In addition we have that $\iota_0(u') = \text{FUTURE}$ for any vertex u' on the path $\phi_0(uv) = \phi(uv)$, and hence $\phi(u') = \text{FUTURE}$ for such u'. It follows that uv is FUTURE-redundant w.r.t σ, the desired contradiction. Essentially the same arguments can also be made for redundant network arcs and redundant tree or network vertices.

- **For any $y, y' \in \{\text{FUTURE}\}$ with $y \neq y'$, there is no arc uv in $D(N_0, T_0)$ for which $\iota_0(u) = y$ and $\iota_0(v) = y'$:** This follows immediately from the fact that $|\{\text{FUTURE}\}| = 1$ so there are no such y, y'.

- **For any u, v, in $V(D(N_0, T_0))$ with $\iota_p(v), \iota_0(v) \in P \cup S$, if $D(N_0, T_0)$ has a path from u to v then $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ has a path from $\iota_0(u)$ to $\iota_0(v)$:** This has already been shown, at the start of the proof for this claim. ⌟

We now have that ψ satisfies all the conditions of a well-behaved F-partial solution. Finally we need to show that σ is the $\{P \to \text{PAST}\}$-restriction of $D(N_0, T_0)$.

Claim 2. *σ is the $\{P \to \text{PAST}\}$-restriction of $D(N_0, T_0)$*

Proof: Let $\sigma'' = (D(N'', T''), \phi'', \iota'')$ denote the $\{P \to \text{PAST}\}$-restriction of ψ. We show $\sigma = \sigma''$ for their three elements individually.

Equality of $D(N'', T'')$ and $D(N, T)$: To show this, it is enough to show that that $D(N'', T'')$ is $D(N_0, T_0)$ with the arcs of A_p and vertices of V_p removed, i.e. these arcs and vertices are exactly the ones that become redundant when constructing the $\{P \to \text{PAST}\}$-restriction of ψ. Since these were exactly

the arcs and vertices that were added to $D(N,T)$ to produce $D(N_0,T_0)$, this is enough to show that $D(N'',T'') = D(N,T)$.

Let g be the restriction function $\{P \to \text{PAST}\}$. We claim that for the isolabelling $\iota_u \circ g$, the redundant arcs are exactly those of A_p.

Indeed, consider any arc uv in A_p. Since $uv \in A_p$, it must have been made PAST-redundant in the construction of σ' from ψ'. So if uv is a tree arc, then $\iota'_0(z) \in P$ for all z in $\{u,v\} \cup V(\phi'_0(uv))$ (as all these vertices are labelled PAST after applying $\{P \to \text{PAST}\}$). Then by construction of ψ, we have $\phi_0(uv) = \phi'_0(uv)$, and also $\iota_0(z) = \iota'_0(z)$ for all z in $\{u,v\} \cup V(\phi'_0(uv))$ (since either $z \in V_p$ or $\iota'(z) = \text{PAST}$). Thus $\iota_0(z) \in P$ for all z in $\{u,v\} \cup V(\phi_0(uv))$, and so uv becomes PAST-redundant after applying $\{P \to \text{PAST}\}$. For a network arc uv in A_p, a similar argument holds, but we need to be careful if there is a tree arc $u'v'$ for which uv is in $\phi(u'v')$ (in particular, we would have a problem if $u'v'$ is not an arc in $D(N',T')$, as $u'v'$ could conceivably prevent uv from becoming PAST-redundant). For such an arc $u'v'$, note that $u'v'$ must also be in A_p (otherwise $u'v'$ is an arc in $D(N',T')$ but $\phi'(uv) = \phi_0(uv)$ is not a path in $D(N',T')$, a contradiction). It follows then that $\iota'_0(u'), \iota_0(v') \in P$. We also have that $\iota'_0(u), \iota_0(v) \in P$. So by a similar argument to tree arcs, we have that uv is redundant after applying $\{P \to \text{PAST}\}$ in ψ, as required.

Conversely consider any redundant arc uv in $D(N_0,T_0)$ after applying $\{P \to \text{PAST}\}$. As ψ is well-behaved, we may assume any such arc is PAST-redundant. Then $\iota_0(u), \iota_0(v) \in P$. This implies among other things that uv is not in A_z (as that would require $\iota_0(u), \iota_0(v) \in F \cup \{\text{FUTURE}\}$) so uv is also an arc in $D(N'_0,T'_0)$. Similarly if uv is a tree arc then all arcs in $\phi_0(uv)$ as also in $D(N'_0,T'_0)$, and if uv is a network arc that is part of a path $\phi_0(u'v')$, then $u'v'$ is also an arc in $D(N_0,T'_0)$. Furthermore $\iota'_0(z) = \iota_0(z)$ for any vertex in one of these arcs (as $\iota'(z) \neq \text{FUTURE}$ and $\iota'(z) \notin V_z$). It is then easy to see that, just as uv is redundant in ψ after applying $\{P \to \text{PAST}\}$, uv is also redundant in ψ' after applying $\{P \to \text{PAST}\}$, and so uv is in A_p.

We have now shown that A_p is exactly the set of arcs in ψ that are redundant after applying $\{P \to \text{PAST}\}$. We now consider the vertices. First consider a vertex $v \in V_p$. Then as V_p is in $D(N'_0,T'_0)$ but not $D(N',T')$, all incident arcs of v are in A_p. Thus v becomes isolated after removing redundant arcs from $D(N,T)$. As $v \in V_p$, by construction $\iota_0(v) = \iota'_0(v) \in P$. Similarly this holds for any v' such that $\phi'_0(v') = v$ or $\phi'_0(v) = v'$. As such, v is PAST-redundant after applying $\{P \to \text{PAST}\}$, as required.

Conversely, suppose v is PAST-redundant w.r.t. $(D(N_0,T_0), \phi_0, \iota_0 \circ \{P \to \text{PAST}\})$. Then v is isolated after removing A_p, and $\iota_0(v), \iota_0(v') \in P$, for v' any vertex such that $\phi_0(v) = v'$ or $\phi_0(v') = v$. These vertices are also in $D(N'_0,T'_0)$ (they cannot be in V_z as that would require them being labelled with something in $\{z, \text{FUTURE}\}$). Then by construction $\iota'_0(v) = \iota_0(v), \iota'_0(v') = \iota_0(v')$ are in P as well, and as such v is PAST-redundant w.r.t $(D(N'_0,T'_0), \phi'_0, \iota'_0 \circ g)$, and so $v \in V_p$, as required.

Equality of ϕ'' and ϕ: Here we use the fact that $D(N'',T'') = D(N,T)$. Consider any arc uv in $T'' = T$. Then by construction, $\phi''(uv) = \phi_0(uv)$, and furthermore $\phi_0(uv) = \phi(uv)$ as uv is an arc in T. Thus $\phi''(uv) = \phi(uv)$ for all arcs uv in $T'' = T$, and so $\phi' = \phi$.

Equality of ι'' and ι: Consider any vertex u in $D(N'',T'') = D(N,T)$, and suppose first that $\iota_0(u) = \iota(u)$. Then $\iota_0(u) \notin P$ (as $\iota(u) \in S \cup \{\text{PAST}, \text{FUTURE}\}$), from which it follows that $\iota''(u) = \iota_0(u) = \iota(u)$. If on the other hand $\iota_0(u) \neq \iota(u)$, by construction of ι_0 this can only happen if $\iota'(u) = \text{PAST}$ (we do not need to consider the case $u \in V_p$ as we know u is a vertex of $D(N,T)$). In this case $\iota_0(u) = \iota'_0(u)$, which must be in P (as otherwise $\iota'(u) = \iota'_0(u) \neq \text{PAST}$.) Then by construction $\iota''(u) = \text{PAST}$, and also $\iota(u) = \iota'(u) = \text{PAST}$. Thus $\iota''(u) = \iota(u)$ for all vertices u in $D(N,T)$. ⌟

As we have now shown that ψ is a well-behaved F-partial solution and σ is the $\{P \to \text{PAST}\}$-restriction of σ, we have that σ is valid, as required. □

3.7 Validity for Join bags

In order to characterize validity for Join bags, we need to introduce a third type of $(S, \mathcal{Y})$-containment structure. In our notation we may think of a Join bag as being expressed by the tuple (P, S, F) where P can be decomposed into $L \cup R$, such that the "left" child bag is $(L, S, F \cup R)$ and the "right" child bag is $(R, S, F \cup L)$. We want to characterize the validity of a signature σ for a Join bag in terms of the validity of signatures for each of the child bags. The main idea is to find two signatures σ_L and σ_R (for different child bags) which are 'compatible', in the sense that the two witnesses for these signatures can be combined, and such that the resulting partial solution is a witness for σ. In order to facilitate this characterization, it will be useful to define a "3-way" analogue of a signature, called a *reconciliation*, in which we use the labels $\{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\}$ instead of $\{\text{PAST}, \text{FUTURE}\}$. This is defined below.

Definition 10 (reconciliation)**.** *Let $(L \cup R, S, F)$ be a Join bag in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$. Then, we call an $(S, \{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\})$-containment structure a* reconciliation *for $(L \cup R, S, F)$. Such a reconciliation μ for $(L \cup R, S, F)$ is called* valid *if it is the $(L \to \text{LEFT}, R \to \text{RIGHT})$-restriction of a well-behaved F-partial solution (i.e. a $(L \cup R \cup S, \{\text{FUTURE}\})$-containment structure).*

Lemma 12. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let σ be a signature for $(L \cup R, S, F)$. Then, σ is valid if and only if there is a reconciliation μ for $(L \cup R, S, F)$ and valid signatures σ_L and σ_R for $(L, S, F \cup R)$ and $(R, S, F \cup L)$, respectively, such that*
(a) σ is the $(\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST})$-restriction of μ,
(b) σ_L is the $(\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE})$-restriction of μ, and
(c) σ_R is the $(\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE})$-restriction of μ.

To prove this lemma, we first show the following.

Lemma 13. *Any valid reconciliation is well-behaved.*

Proof: Let μ be a valid reconciliation for $(L \cup R, S, F)$, and let $\psi = (D(N, T), \phi, \iota)$ be a well-behaved F-partial solution for which μ is the g-restriction, where g is the function $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$. We first show that there is no arc uv in $D(N, T)$ with $g(\iota(u)) = y, g(\iota(v)) = y'$ for any $y, y' \in \{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\}$ with $y \neq y'$. Recall that by properties of a tree decomposition, there are no arcs between F and $L \cup R$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, nor between L and R. So now consider a vertex $u \in V(D(N, T))$; we will show that if $g(\iota(u)) \in \{\text{LEFT}, \text{RIGHT}\}$ then u has no neighbour v with $g(\iota(v)) \in \{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\} \setminus \{g(\iota(u))\}$, which is enough to show the claim. If $\iota(u) \in L$, then all neighbours of $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ are in $L \cup S$. Furthermore as ψ is a well-behaved $(L \cup R \cup S, \{\text{FUTURE}\})$-containment structure, the degree of u in $D(N, T)$ is equal to the degree of $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, and for each neighbour v' of $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, there is a neighbour v of u in $D(N, T)$ with $\iota(v) = v'$. It follows that $\iota(v) \in L \cup S$ and thus $g(\iota(v)) \in S \cup \{\text{LEFT}\}$ for any neighbour of u in $D(N, T)$, while $g(\iota(u)) = \text{LEFT}$. A similar argument shows that if $\iota(u) \in R$, then $g(\iota(v)) \in S \cup \{\text{RIGHT}\}$ for any neighbour of u in $D(N, T)$ and $g(\iota(u)) = \text{RIGHT}$. Finally if $\iota(u) \in S \cup \{\text{FUTURE}\}$, then $g(\iota(u)) = \iota(u) \notin \{\text{LEFT}, \text{RIGHT}\}$, and we are done.

As no vertex labelled LEFT or RIGHT by $\iota \circ g$ has a neighbour with a different label from $\{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\}$, there are in fact no arcs uv in $D(N,T)$ with $g(\iota(u)) = y, g(\iota(v)) = y'$ for any $y, y' \in \{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\}$ with $y \neq y'$. We can therefore apply Lemma 6 to see that σ is well-behaved. □

We next motivate the definition of a reconciliation by showing that the validity of a signature for $(L \cup R, S, F)$ can be characterized by the validity of reconciliations for $(L \cup R, S, F)$.

Lemma 14. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let σ be a signature for $(L \cup R, S, F)$. Then σ is valid if and only if there is a valid reconciliation μ for $(L \cup R, S, F)$ such that σ is the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ.*

Proof: Suppose first that σ is valid. Then there is a well-behaved $(L \cup R \cup S, \{\text{FUTURE}\})$-containment structure ψ (an F-partial solution) such that σ is the $\{L \cup R \to \text{PAST}\}$-restriction of ψ. Now let μ be the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ. By Lemma 3, μ is an $(S, \{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\})$-containment structure, and by construction μ is valid. Now let σ' be the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ. Then transitivity implies that σ' is also the $\{L \cup R \to \text{PAST}\}$-restriction of ψ, that is $\sigma' = \sigma$. Thus σ is the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ, as required.

Conversely, suppose there is a reconciliation μ for $(L \cup R, S, F)$ such that σ is the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ. Then as μ is valid, there is a well-behaved F-partial solution ψ such that μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ. Then again by transitivity, σ is also the $\{L \cup R \to \text{PAST}\}$-restriction of ψ. Thus σ is valid. □

Now we show how the validity of a reconciliation μ for $(L \cup R, S, F)$ can be characterized by the validity of signatures for the child bags.

Lemma 15. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let μ be a well-behaved reconciliation for $(L \cup R, S, F)$. Let σ_L be the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and σ_R the $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ. If μ is valid, then σ_L is a valid signature for $(L, S, F \cup R)$ and σ_R is a valid signature for $(R, S, F \cup L)$.*

Proof: Suppose that μ is valid. Then there is a well-behaved F-partial solution ψ such that μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of μ. By construction and Lemma 3, σ_L is an $(S, \{\text{PAST}, \text{FUTURE}\})$-containment structure, and thus a signature for $(L, S, F \cup R)$. By transitivity, σ_L is the $\{L \to \text{PAST}, R \to \text{FUTURE}\}$-restriction of ψ. We will show that σ_L is a valid signature for $(L, S, F \cup R)$.

Let ψ_L be the $\{R \to \text{FUTURE}\}$-restriction of ψ. By construction, ψ_L is an $F \cup R$-partial solution and by Lemma 6 ψ_L is well-behaved. Moreover by transitivity, the $\{L \to \text{PAST}\}$-restriction of ψ_L is also the $\{L \to \text{PAST}, R \to \text{FUTURE}\}$-restriction of ψ. That is, the $\{L \to \text{PAST}\}$-restriction of ψ_L is σ_L, and so σ_L is valid for $(L, S, F \cup R)$. (See Fig. 7.)

A similar argument shows that σ_R is a valid signature for $(R, S, F \cup R)$. □

Lemma 16. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let μ be a well-behaved reconciliation for $(L \cup R, S, F)$. Let σ_L be the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and σ_R the $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ. If σ_L is a valid signature for $(L, S, F \cup R)$ and σ_R is a valid signature for $(R, S, F \cup L)$, then μ is valid.*

Proof: Let ψ_L be a well-behaved $F \cup R$-partial solution for which σ_L is the $(L \to \text{PAST})$-restriction, and similarly let ψ_R be a well-behaved $F \cup L$-partial solution for which σ_R is the $(R \to \text{PAST})$-restriction.

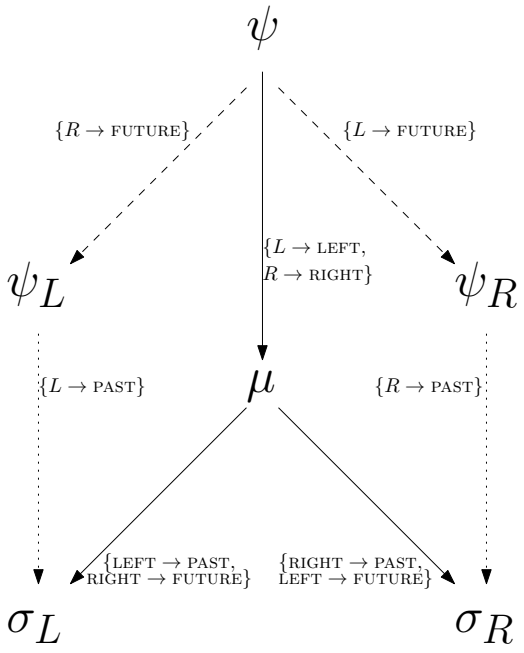

Figure 7: Illustration of proof for reconciliations on Join bags, validity of μ implies validity of σ_L and σ_R. Solid lines are restriction relations that we may assume; the dashed lines shows the construction of ψ_L and ψ_R from ψ; dotted lines are relations we can infer using (multiple uses of) transitivity.

Our strategy is to combine ψ_L and ψ_R into an F-partial solution ψ, and then show that μ is the $(L \to \text{LEFT}, R \to \text{RIGHT})$-restriction of this ψ.

In what follows, for a given containment structure σ, we will write the display graph, embedding function and isolabelling of σ as $D(N_\sigma, T_\sigma), \phi_\sigma$ and ι_σ. Thus we have $\mu = (D(N_\mu, T_\mu), \phi_\mu, \iota_\mu)$, $\sigma_L = (D(N_{\sigma_L}, T_{\sigma_L}), \phi_{\sigma_L}, \iota_{\sigma_L})$, etc. Recall that by construction, $D(N_{\sigma_L}, T_{\sigma_L})$ is a subgraph of both $D(N_\mu, T_\mu)$ and $D(N_{\psi_L}, T_{\psi_L})$. Similarly, $D(N_{\sigma_R}, T_{\sigma_R})$ is a subgraph of both $D(N_\mu, T_\mu)$ and $D(N_{\psi_R}, T_{\psi_R})$.

Now let A_L be the set of arcs, and V_L the set of vertices, that become redundant and are therefore deleted from $D(N_{\psi_L}, T_{\psi_L})$ when deriving the $\{L \to \text{PAST}\}$-restriction σ_L from ψ_L. That is, $A_L = A(D(N_{\psi_L}, T_{\psi_L})) \setminus A(D(N_{\sigma_L}, T_{\sigma_L}))$ and $V_L = V(D(N_{\psi_L}, T_{\psi_L})) \setminus V(D(N_{\sigma_L}, T_{\sigma_L}))$. We note that by construction, $\iota_{\psi_L}(v) \in L$ for any $v \in V_L \cup V(A_L)$. Similarly, let A_R, V_R be the sets of arcs and vertices that are deleted from $D(N_{\psi_R}, T_{\psi_R})$ in the construction of σ_R from ψ_R. Let A'_R, V'_R be the set of vertices that are deleted from $D(N_\mu, T_\mu)$ in the construction of σ_L from μ, and let A'_L, V'_L be the set of vertices that are deleted from $D(N_\mu, T_\mu)$ in the construction of σ_R from μ. (See Fig. 8.)

Finally let A_S be the arcs of $D(N_\mu, T_\mu)$ that are not in $A'_L \cup A'_R$, and V_S the vertices of $D(N_\mu, T_\mu)$ not in $V'_L \cup V'_R$. Note that the arcs and vertices of V_S, A_S appear in all of $D(N_\mu, T_\mu)$, $D(N_{\sigma_L}, T_{\sigma_L})$, $D(N_{\sigma_R}, T_{\sigma_R})$, $D(N_{\psi_L}, T_{\psi_L})$, $D(N_{\psi_R}, T_{\psi_R})$.

Observe that if $v \in V'_R \cup V(A'_R)$ then $\iota_\mu(v) \in \{\text{RIGHT}, \text{FUTURE}\}$. Indeed, when deriving σ_L the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, no arcs or vertices will become PAST-redundant, since they would previously have been LEFT-redundant in μ (here we use the fact that ι_μ did not already

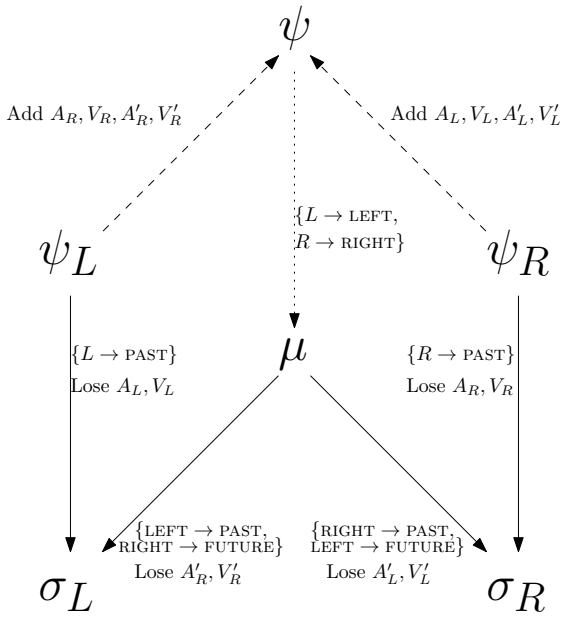

Figure 8: Illustration of proof for reconciliations on Join bags, validity of σ_L and σ_R implies validity of μ. Solid lines are restriction relations that we may assume; the dashed lines show the construction of ψ from ψ_L or ψ_R (we do not describe the construction of ϕ_0 or ι_0 in the figure, only the construction of $D(N_0, T_0)$ by adding arcs and vertices); the dotted line shows the relation we want to prove, that μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ.

label any vertices PAST, unlike FUTURE.) So the only vertices and arcs that are removed are ones that become FUTURE-redundant, i.e. previously had their vertices labelled RIGHT or FUTURE. By a similar argument, if $v \in V'_L \cup V(A'_L)$ then $\iota_\mu(v) \in \{\text{LEFT}, \text{FUTURE}\}$. Furthermore, we can show that $A'_L \cap A'_R = \varnothing$ and $V'_L \cap V'_R = \varnothing$. Indeed, recall that for any arc a in $D(N_\mu, T_\mu)$, there is a set of vertices Q_a such that a is y-redundant w.r.t $(D(N_\mu, T_\mu), \phi_\mu, \iota')$ if and only if $\iota(Q_a) = \{y\}$, for any isolabelling ι. Note that we cannot have $\iota_\mu(Q_a) = \{\text{FUTURE}\}$, as this implies that a is redundant w.r.t μ, a contradiction as μ is well-behaved. So if a is FUTURE-redundant after applying $\{L \to \text{PAST}, R \to \text{FUTURE}\}$ (i.e. if $a \in A'_R$) then there is at least one $z \in Q_a$ with $\iota_\mu(z) \in R$. But then this implies that a is not FUTURE-redundant after applying $\{R \to \text{PAST}, L \to \text{FUTURE}\}$, so a is not in A'_L. A similar argument shows that V'_L and V'_R are disjoint.

We now describe the construction of an F-partial solution $\psi = (D(N, T), \phi, \iota)$.

Let $D(N, T)$ be the display graph with vertex set $V_S \cup V'_L \cup V'_R \cup V_L \cup V_R$ and arc set $A_S \cup A'_L \cup A'_R \cup A_L \cup A_R$. (We keep the partition of these sets into network side and tree side the same as before.) That is, $D(N, T)$ is $D(N_{\psi_L}, T_{\psi_L})$ with the arcs of $A_R \cup A'_R$ and vertices of $V_R \cup V'_R$ added; equivalently we may say $D(N, V)$ is $D(N_{\psi_R}, T_{\psi_R})$ with the arcs of $A_L \cup A'_L$ and vertices of $V_L \cup V'_L$ added, or that it is $D(N_\mu, T_\mu)$ with the arcs of $A_L \cup A_R$ and vertices of $V_L \cup V_R$ added.

Let the embedding function ϕ be defined as follows. For a tree vertex u in T, if $u \in V_L$ then let $\phi(u) = \phi_{\psi_L}(u)$, and similarly if $u \in V_R$ then let $\phi(u) = \phi_{\psi_R}(u)$. If $u \in V'_L$, let $\phi(u) = \phi_\mu(u) =$

$\phi_{\sigma_L}(u) = \phi_{\psi_L}(u)$ (note that u is a vertex in $D(N_{\sigma_L}, T_{\sigma_L})$ as $v \notin V'_R$, and so these terms are all well-defined and equal by construction). Similarly if $u \in V'_R$, let $\phi(u) = \phi_\mu(u) = \phi_{\sigma_R}(u) = \phi_{\psi_R}(u)$. Finally if $u \in V'_S$, let $\phi(u) = \phi_\mu(u) = \phi_{\sigma_L}(u) = \phi_{\sigma_R}(u) = \phi_{\psi_L}(u) = \phi_{\psi_R}(u)$.

For a tree arc uv in A_L, let $\phi(uv) = \phi_{\psi_L}(uv)$, and similarly for $uv \in A_R$ let $\phi(uv) = \phi_{\psi_R}(uv)$. For $uv \in A'_L$, let $\phi(uv) = \phi_\mu(uv) = \phi_{\sigma_L}(uv) = \phi_{\psi_L}(uv)$ (note that uv is an arc in $D(N_{\sigma_L}, T_{\sigma_L})$ as $uv \notin A'_R$). Similarly for $uv \in A'_R$, let $\phi(uv) = \phi_\mu(uv) = \phi_{\sigma_R}(uv) = \phi_{\psi_R}(uv)$. Finally for $uv \in A_S$, let $\phi(uv) = \phi_\mu(uv) = \phi_{\sigma_L}(uv) = \phi_{\sigma_R}(uv) = \phi_{\psi_L}(uv) = \phi_{\psi_R}(uv)$.

Let the $(L \cup R \cup S, \{\text{FUTURE}\})$-isolabelling ι be defined as follows. For a vertex $v \in V_L$, let $\iota(v) = \iota_{\psi_L}(v)$. Similarly if $v \in V_R$, let $\iota(v) = \iota_{\psi_R}(v)$. Otherwise, v is a vertex of $D(N_\mu, T_\mu)$. If $\iota_\mu(v) = \text{LEFT}$ then let $\iota(v) = \iota_{\psi_L}(v)$ (recalling that $v \notin V'_R$, so v is a vertex of $D(N_{\sigma_L}, T_{\sigma_L})$ and thus of $D(N_{\psi_L}, T_{\psi_L})$, with $\iota_{\sigma_L}(v) = \text{PAST}$ and $\iota_{\psi_L}(v) \in L$). Similarly if $\iota_\mu(v) = \text{RIGHT}$ then let $\iota(v) = \iota_{\psi_R}(v)$. Finally if $\iota_\mu(v) \in S \cup \{\text{FUTURE}\}$, then set $\iota(v) = \iota_\mu(v)$.

We will now show that ψ is indeed a well-behaved F-partial solution; afterwards we will show that μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ, from which it follows that μ is valid. (A similar argument can be used to show that ψ_L is the $\{R \to \text{FUTURE}\}$-restriction of ψ and ψ_R is the $\{L \to \text{FUTURE}\}$-restriction of ψ, but we will not need these fact so we do not prove them here.)

Claim 3. *ψ is a well-behaved F-partial solution.*

Proof: We first show that ψ satisfies one of the properties of well-behaved $(L \cup R \cup S, \{\text{FUTURE}\})$-containment structure.

- **For any $u, v \in V(D(N,T))$ with $\iota(u), \iota(v) \in L \cup R \cup S$, if $D(N,T)$ has a path from u to v then there is a path from $\iota(u)$ to $\iota(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$:** To see this property, first consider the case that uv is an arc in $D(N,T)$. In this case it is sufficient to show that $\iota(u) = \iota_\theta(u)$ and $\iota(v) = \iota_\theta(v)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$, as the property then follows from the fact that μ, ψ_L, ψ_R are all well-behaved.

  If $uv \in A_L$ then $u \in V(D(N_{\psi_L}, T_{\psi_L}))$ and either $u \in V_L$ or $\iota_{\sigma_L}(u) = \text{PAST}$ and $\iota_\mu(u) = \text{LEFT}$; in either case $\iota(u) = \iota_{\psi_L}(u)$, and similarly $\iota(v) = \iota_{\psi_L}(v)$, so we let $\theta = \psi_L$. Similarly if $uv \in A_R$ then $\theta = \psi_R$. For any arc uv in $D(N_\mu, T_\mu)$, we can let $\theta = \psi_L$ if $\iota_\mu(u) = \iota_\mu(v) = \text{LEFT}$, ψ_R if $\iota_\mu(u) = \iota_\mu(v) = \text{RIGHT}$, and μ if $\iota_\mu(u), \iota_\mu(v) \in P$. If $\iota_\mu(u) = \text{FUTURE}$ (resp. $\iota_\mu(v) = \text{FUTURE}$) then $\iota(u) = \text{FUTURE}$ ($\iota(v) = \text{FUTURE}$), so we do not need to consider this case. We also cannot have $\{\iota_\mu(u), \iota_\mu(v)\} = \{\text{LEFT}, \text{RIGHT}\}$ as μ is well-behaved. It remains to consider the case that one of $\iota_\mu(u), \iota_\mu(v)$ is LEFT or RIGHT and the other is in S; suppose w.l.o.g. that $\iota_\mu(u) = \text{LEFT}$ and $\iota_\mu(v) \in S$. In this case both u and v are vertices of $D(N_{\sigma_L}, T_{\sigma_L})$ and therefore $D(N_{\psi_L}, T_{\psi_L})$; moreover $\iota(u) = \iota_{\psi_L}(u)$ and $\iota_{\psi_L}(v) = \iota_{\sigma_L}(v) = \iota_\mu(v) = \iota(v)$. Thus we can let $\theta = \sigma_L$.

  Now suppose for a contradiction that there exist $u, v \in V(D(N,T))$ with $\iota(u), \iota(v) \in L \cup R \cup S$ such that there is a path from u to v in $D(N,T)$, but no path from $\iota(u)$ to $\iota(v)$ exists in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. By the above discussion uv cannot be an arc. Moreover every internal vertex z on the shortest $u - v$ path must have $\iota(z) = \text{FUTURE}$ (otherwise $\iota(z) \in L \cup R \cup S$ and u, z forms a shorter path). Note that any such z is also a vertex in $D(N_\mu, T_\mu)$ with $\iota_\mu(z) = \text{FUTURE}$ (no other possibility leads to $\iota(z) = \text{FUTURE}$; in particular if $\iota_\mu(z) = \text{LEFT}$ then $\iota(z) = \iota_{\psi_L}(z) \in L$). Any arc incident to z in $D(N,T)$ is also an arc in $D(N_\mu, T_\mu)$ (such arcs cannot be in A_L or A_R). So it follows that the path from u to v also exists in $D(N_\mu, T_\mu)$. Finally, we must have $\iota_\mu(u), \iota_\mu(v) \in S$ (they cannot be in $\{\text{LEFT}, \text{RIGHT}\}$ as u, v have neighbours labelled FUTURE by ι_μ and μ is well-behaved). So

it then follows that there is a path from $\iota(u) = \iota_\mu(u)$ to $\iota(v) = \iota_\mu(v)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, as μ is well-behaved.

- $D(N,T)$ **is a display graph:**
  - $D(N,T)$ **is acyclic:** Suppose for a contradiction that $D(N,T)$ contains a cycle. If $\iota(u) \in L \cup R \cup S$ for some vertex u in this cycle, then as shown above the existence of a path from u to u in $D(N,T)$ implies the existence of a path from $\iota(u)$ to $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, contradicting the fact that $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ is acyclic.

    It remains to consider the case that every vertex z in this cycle has $\iota(z) = \text{FUTURE}$. However as argued previously, any z with $\iota(z) = \text{FUTURE}$ is also a vertex in $D(N_\mu, T_\mu)$, and all its incident arcs in $D(N,T)$ are also arcs of $D(N_\mu, T_\mu)$. Then we have that $D(N_\mu, T_\mu)$ contains a cycle, a contradiction as $D(N_\mu, T_\mu)$ is a display graph.
  - T **is an out-forest:** As $D(N,T)$ is acyclic, it is remains to show that every vertex of T has in-degree at most 1 in T. To do this, we will show something stronger: that for any vertex v in $D(N,T)$, all of its incident arcs in $D(N,T)$ are arcs in $D(N_\theta, T_\theta)$, for some $\theta \in \{\mu, \psi_L, \psi_R\}$. Then the desired property immediately follows, as every tree vertex has at most one incoming tree-arc in $D(N_\theta, T_\theta)$.

    So consider any vertex $v \in D(N,T)$. If $v \in V_L$, then the only incident arcs of v are in A_L, and therefore all these arcs in $D(N_{\psi_L}, T_{\psi_L})$. Similarly if $v \in V_R$ then all incident arcs are in $D(N_{\psi_R}, T_{\psi_R})$. So now we may assume v is a vertex of $D(N_\mu, T_\mu)$. If $\iota_\mu(v) = \text{LEFT}$ then v can have no incident arcs in A'_R or A_R (as any vertex incident to such an arc must be in V_R or else have $\iota_{\sigma_R}(u) = \text{PAST}$ and so $\iota_\mu(v) = \text{RIGHT}$). Then all incident arcs of uv in $D(N,T)$ are also in $D(N_{\psi_L}, T_{\psi_L})$. Similarly if $\iota_\mu(v) = \text{LEFT}$ then all incident arcs are in $D(N_{\psi_R}, T_{\psi_R})$. Finally if $\iota_\mu(v) \in S \cup \{\text{FUTURE}\}$, then again none of its incident arcs are in A_L or A_R (as this would require $\iota_{\sigma_L}(v) = \text{PAST}$ or $\iota_{\sigma_R}(v) = \text{PAST}$ and so $\iota_\mu(v) \in \{\text{LEFT}, \text{RIGHT}\}$), and so so all incident arcs are in $D(N_\mu, T_\mu)$.
  - **Every vertex in** $D(N,T)$ **has in- and out-degree at most** 2 **and total degree at most** 3**:** This follows immediately from the previously-shown property that every vertex in $D(N,T)$ has all its incident arcs in one of $D(N_\mu, T_\mu), D(N_{\psi_L}, T_{\psi_L}), D(N_{\psi_R}, T_{\psi_R})$, and the fact that this constraint holds for each of these graphs.
  - **Any vertex in** $V(N) \cap V(T)$ **has out-degree** 0 **and in-degree at most** 1 **in each of** T_0 **and** N_0**:** Again this follows from the fact that every vertex in $D(N,T)$ has all its incident arcs in one of $D(N_\mu, T_\mu), D(N_{\psi_L}, T_{\psi_L}), D(N_{\psi_R}, T_{\psi_R})$.
- ϕ **is an embedding function on** $D(N,T)$**:**
  - **For each** $u \in V(T)$**,** $\phi(u) \in V(N)$ **and, for each arc** $uv \in A(T)$**,** $\phi(uv)$ **is a directed** $\phi(u)$**-**$\phi(v)$**-path in** N**:**

    The fact that $\phi(u) \in V(N)$ follows immediately from the fact that $\phi(u) = \phi_\theta(u)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$, and so $\phi(u)$ is a network vertex.

    To see that $\phi(uv)$ is a path from $\phi(u)$ to $\phi(v)$, we will show that there is $\theta \in \{\mu, \psi_L, \psi_R\}$ such that $\phi(uv) = \phi_\theta(uv)$, $\phi(u) = \phi_\theta(u)$ and $\phi(v) = \phi_\theta(v)$. The result then follows from the fact that ϕ_θ is an embedding function.

If $uv \in A_L$, then neither u nor v can be in V_R, nor can they be in V'_R (note that $u \in V'_R$ requires $\iota_\mu(u) \in \{\text{RIGHT}, \text{FUTURE}\}$, but $uv \in A_L$ implies either $u \in V_L$ or $\iota_{\sigma_L}(u) = \text{PAST}$ and thus $\iota_\mu(u) = \text{LEFT}$). It follows by construction of ϕ that $\phi(u) = \phi_{\psi_L}(u)$, $\phi(v) = \phi_{\psi_L}(u)$, and $\phi(uv) = \phi_{\psi_L}(uv)$, so we can let $\theta = \psi_L$. Similarly if $uv \in A_R$ we can let $\theta = \psi_R$. For $uv \in A'_L \cup A'_R \cup A_S$, we have that u, v are vertices in $D(N_\mu, T_\mu)$, and therefore not in V_L or V_R. It follows by construction that $\phi(u) = \phi_\mu(u)$, $\phi(v) = \phi_\mu(u)$, and $\phi(uv) = \phi_\mu(uv)$, so we let $\theta = \mu$.

- **for any distinct** $u, v \in V(T)$**,** $\phi(u) \neq \phi(v)$: We first show that for any $u \in V(T)$, $u \in V_L$ if and only if $\phi(u) \in V_L$. Recall that V_L is the set of vertices in $D(N_{\psi_L}, T_{\psi_L})$ that become PAST-redundant after applying $\{L \to \text{PAST}\}$, and observe that by the definition of y-redundancy, a tree vertex u is y-redundant with respect to some containment structure $(D(N', T'), \phi', \iota')$ if and only if $\phi'(u)$ is y-redundant. Thus $u \in V_L$ if and only if $\phi(u) \in V_L$. By a similar argument, $u \in V_R$ if and only if $\phi(u) \in V_R$.

  Now suppose for a contradiction that there exist distinct $u, v \in V(T)$ with $\phi(u) = \phi(v)$. If $\phi(u) = \phi(v) \in V_L$, then also $u, v \in V_L$. Thus u, v are distinct vertices in $D(N_{\psi_L}, T_{\psi_L})$ with $\phi_{\psi_L}(u) = \phi(u) = \phi(v) = \phi_{\psi_R}(v)$, a contradiction as ϕ_{ψ_L} is an embedding function. We get a similar contradiction if $\phi(u) \in V_R$. Finally if $\phi(u) \notin V_L \cup V_R$, then also $u, v \notin V_L \cup V_R$. Thus u, v are distinct vertices in $D(N_\mu, T_\mu)$ with $\phi_\mu(u) = \phi(u) = \phi(v) = \phi_\mu(v)$, again a contradiction as ϕ_μ is an embedding function.
- **for any** $u \in V(T) \cap V(N)$**,** $\phi(u) = u$**:** This follows immediately from the fact that u is a vertex in $V(T_\theta) \cap V(N_\theta)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$, and for such a θ $\phi(u) = \phi_\theta(u) = u$.
- **the paths** $\{\phi(uv) \mid uv \in A(T)\}$ **are arc-disjoint:**
  Similar to the proof that $\phi(u) \neq \phi(v)$ for $u \neq v$, we observe that if an arc uv is y-redundant with respect to some $(D(N', T'), \phi', \iota')$, then so are all the arcs of $\phi'(uv)$. It follows by construction that if $uv \in A_L$ (resp. A_R, AL', A'_R, A_S) then so are all arcs of $\phi(uv)$. So now suppose for a contradiction that there exist distinct tree arcs uv, $u'v'$ such that $\phi(uv), \phi(u'v')$ share an arc. Then uv and $u'v'$ are both in the same set from $\{A_L, A_R, A'_L, A'_R, A_S\}$, and so in particular they are both in $D(N_\theta, T_\theta)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$, with $\phi(uv) = \phi_\theta(uv)$, $\phi(u'v') = \phi_\theta(u'v')$. Then $\phi_\theta(uv), \phi_\theta(u'v')$ share an arc, a contradiction as ϕ_θ is an embedding function.
- **for any distinct** $p, q \in A(T)$**,** $\phi(p)$ **and** $\phi(q)$ **share a vertex** z **only if** p **and** q **share a vertex** w **with** $z = \phi(w)$**:** Here we use a couple of properties that have been proved earlier. Recall from the proof that T is an out-forest, that for any vertex v in $D(N, T)$, all its incident arcs in $D(N, T)$ are arcs in $D(N_\theta, T_\theta)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$. Recall also, from the proof that the paths $\{\phi(uv) | uv \in A(T)\}$ are arc disjoint, that a tree arc uv is in A_L (resp. A_R, A'_L, A'_R, A_S) then so are all arcs in $\phi(uv)$.

  So now consider distinct arcs $p, q \in A(T)$ and suppose $\phi(p)$ and $\phi(q)$ share a vertex z. Suppose all incdent arcs of z are in $D(N_\mu, T_\mu)$ (the cases where all incident arcs are in $D(N_{\psi_L}, T_{\psi_L})$ or $D(N_{\psi_R}, T_{\psi_R})$ are similar). Then all incident arcs of z from $\phi(p)$ and $\phi(q)$ are in $A'_L \cup A'_R \cup A_S$, which implies that p, q are in $A'_L \cup A'_R \cup A_S$ as well. Thus p, q are both arcs in $D(N_\mu, T_\mu)$. It follows that $\phi(p) = \phi_\mu(p)$ and $\phi(q) = \phi_\mu(q)$ also share the vertex z, which implies that p, q share a vertex w with $z = \phi_\mu(w)$, as ϕ_μ is an embedding function. Finally observe that $w \in V'_L \cup V'_R \cup V_S$ (as w is in $D(N_\mu, T_\mu)$) so $\phi(w) = \phi_\mu(v) = z$, as required.

- ι **is a** $(L \cup R \cup S, \{\textsc{future}\})$**-isolabelling:**
  - **For** $u \in V(D(N,T))$ **with** $\iota(u) \neq \textsc{future}$**,** $\iota(u) \in V(N_{\text{IN}})$ **only if** $u \in V(N)$ **and** $\iota(u) \in V(T_{\text{IN}})$ **only if** $u \in V(T)$ **:** This follows from the fact that $\iota(u) = \iota_\theta(u)$ for some $\theta \in \{\mu, \psi_L, \psi_R\}$.
  - **For** $u, v \in V(D(N,T))$ **with** $\iota(u), \iota(v) \neq \textsc{future}$**,** $\iota(u) = \iota(v)$ **only if** $u = v$ **:**
    Consider some $u, v \in V(D(N,T))$ with $\iota(u) = \iota(v) \neq$ FUTURE. If $\iota(u) \in L$, then $\iota(u) = \iota_{\psi_L}(u)$, and $\iota(v) = \iota_{\psi_L}(v)$, as of the isolabelings $\iota_{\psi_L}, \iota_{\psi_R}, \iota_\mu$, only ι_{ψ_L} maps anything to L. Thus $\iota_{\psi_L}(u) = \iota_{\psi_L}(v) \in L$, from which it follows that $u = v$ since ψ_L is an isolabelling. Similarly, if $\iota(u) \in R$ then $\iota_{\psi_R}(u) = \iota_{\psi_R}(v) \in R$ and so $u = v$. If $\iota(u) \in S$, then u and v must be vertices of $D(N_\mu, T_\mu)$ with $\iota(u) = \iota_\mu(u)$ and $\iota(v) = \iota_\mu(v)$ (other possibilities, such as $u \in V_L$ or $\iota_\mu(u) =$ RIGHT, imply $\iota(u)$ or $\iota(v)$ are in in $L \cup R$). Then again as ι_μ is an isolabelling, we have $u = v$.
  - **For** $u, v \in V(D(N,T))$ **with** $\iota(u), \iota(v) \neq \textsc{future}$**, the arc** uv **is in** $D(N,T)$ **if and only if** $\iota(u)\iota(v)$ **is in** $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ **:**
    First suppose that uv is an arc in $D(N,T)$, for $\iota(u), \iota(v) \neq$ FUTURE. We aim to show that there is $\theta \in \{\psi_L, \psi_R, \mu\}$ such that uv is an arc in $D(N_\theta, T_\theta)$ and $\iota(u) = \iota_\theta(u)$, $\iota(v) = \iota_\theta(v)$. Then $\iota(u)\iota(v) \in A(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ follows from the fact that θ is well-behaved.
    If $uv \in A_L$ then uv is an arc of $D(N_{\psi_L}, T_{\psi_L})$, and either $u \in V_L$ or $\iota_\mu(u) =$ LEFT, which implies $\iota(u) = \iota_{\psi_L}(u)$. Similarly $\iota(v) = \iota_{\psi_L}(v)$, so we can let $\theta = \psi_L$. By a similar argument, if $uv \in A_R$ then we let $\theta = \psi_R$. For $uv \in A'_L$, we again have that uv is an arc of $D(N_{\psi_L}, T_{\psi_L})$, and $\iota_\mu(u) =$ LEFT, which implies $\iota(u) = \iota_{\psi_L}(u)$. Similarly $\iota(v) = \iota_{\psi_L}(v)$, so we can let $\theta = \psi_L$. By a similar argument, if $uv \in A'_R$ then we let $\theta = \psi_R$. For $uv \in A_S$, we make use of the fact that $\{\iota_\mu(u), \iota_\mu(v)\} \neq \{$LEFT, RIGHT$\}$ (as μ is well-behaved). So suppose w.l.o.g that $\iota_\mu(u), \iota_\mu(v) \neq$ RIGHT. Then either $\iota_\mu(u) \in S$, in which case $\iota_{\psi_L}(u) = \iota_\mu(u) = \iota(u)$, or $\iota_\mu(u) =$ LEFT, in which case $\iota(u) = \iota_{\psi_L}(u)$. Thus in either case $\iota(u) = \iota_{\psi_L}(u)$, and similarly $\iota(v) = \iota_{\psi_L}(v)$. Note also that $uv \in A(D(N_{\psi_L}, T_{\psi_L}))$. Thus we can let $\theta = \psi_L$, as required.
    For the converse, suppose that $\iota(u)\iota(v)$ is an arc in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. By the properties of a tree decomposition, we cannot have $\iota(u) \in L$ and $\iota(v) \in R$ (or vice-versa). If $\iota(u), \iota(v) \in L$ then it must hold that $\iota(u) = \iota_{\psi_L}(u)$ and $\iota(v) = \iota_{\psi_L}(v)$. Then as ψ_L is well-behaved, uv is an arc in $D(N_{\psi_L}, T_{\psi_L})$, which also implies that uv is an arc in $D(N,T)$. A similar argument holds if $\iota(u), \iota(v) \in R$. If $\iota(u), \iota(v) \in S$ then it must hold that $\iota(u) = \iota_\mu(u)$ and $\iota(v) = \iota_\mu(v)$. Then again as μ is well-behaved, uv is an arc in $D(N_\mu, T_\mu)$ and thus $D(N,T)$. It remains to consider the case where one of $\iota(u), \iota(v)$ is in S and the other is in L or R. Suppose w.l.o.g. that $\iota(u) \in L, \iota(v) \in S$. Then $\iota(u) = \iota_{\psi_L}(u)$, and $\iota(v) = \iota_\mu(v)$, seemingly a problem. However, notice that as $\iota_\mu(v) \in S$, we also have that v is a vertex of $D(N_{\sigma_L}, T_{\sigma_L})$ and $D(N_{\psi_L}, T_{\psi_L})$, with $\iota_{\psi_L}(v) = \iota_{\sigma_L}(v) = \iota_\mu(v) = \iota(v)$. Thus $\iota_{\psi_L}(u)\iota_\psi(v)$ is an arc in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, from which it follows that uv is an arc in $D(N_{\psi_L}, T_{\psi_L})$ and thus $D(N,T)$.
  - $\iota(V(D(N,T))) \cap V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) = L \cup R \cup S$ **:** By construction, $\iota(v) \in L \cup R \cup S \cup \{\textsc{future}\}$ for any $v \in V(D,T)$, so $\iota(V(D(N,T))) \cap V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \subseteq L \cup R \cup S$. It remains to show that for any $z \in L \cup R \cup S$, there is $v \in V(D(N,T))$ for which $\iota(v) = z$. If $z \in L$, then as ι_{ψ_L} is a $(L \cup S, \{\textsc{future}\})$-isolabelling, there is $v \in V(D(N_{\psi_L}, T_{\psi_L}))$ such that $\iota_{\psi_L}(v) = z$. Then either $v \in V_L$, or v is a vertex of $D(N_{\sigma_L}, T_{\sigma_L})$ and $D(N_\mu, T_\mu)$ with

$\iota_{\sigma_L}(v) = \text{PAST}$, $\iota_\mu(v) = \text{LEFT}$ (since σ_L is the $\{L \to \text{PAST}\}$-restriction of ψ_L). In either case $\psi(v) = \psi_L(v)$, and so $\psi(v) = z$ as required. By a similar argument, if $z \in R$ then there is $v \in V(D(N,T))$ with $\iota(v) = \iota_{\psi_L}(v) = z$. Finally if $z \in S$, there is a vertex v in $D(N_\mu, T_\mu)$ with $\iota_\mu(v) = z$. As $\iota_\mu(v) \notin \{\text{LEFT}, \text{RIGHT}\}$, by construction we have $\iota(v) = \iota_\mu(v) = z$, as required.

- **Each vertex u with $\iota(u) \in L \cup R \cup S$ has the same in- and out-degree in $D(N,T)$ as $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$:** Recall, from the proof that T is an out-forest, that any vertex v in $D(N,T)$ has all its incident vertices contained in $D(N_\theta, T_\theta)$, for some $\theta \in \{\psi_L, \psi_R, \mu\}$. Moreover if $u \in V_L$ or $\iota_\mu(u) = \text{LEFT}$ (in which case $\psi(u) = \psi_L(u)$ by construction) then all incident arcs of u are in $D(N_{\psi_L}, T_{\psi_L})$. Thus u has the same in- and out-degree in $D(N,T)$ as $\iota(u) = \iota_{\psi_L}(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, since ψ_L is a containment structure. A similar argument holds if $u \in V_R$ or $\iota_\mu(u) = \text{RIGHT}$. Finally if $\iota_\mu(u) \in S$, then all incident arcs of u are in $D(N_\mu, T_\mu)$, and so u has the same degree in $D(N,T)$ as $\iota(u) = \iota_\mu(u)$ does in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$.

- **Each vertex u of T with $\iota(u) \neq \iota(\phi(u))$ has 2 out-arcs in $D(N,T)$:**

  In the case that $\iota(u) \in L \cup R \cup S$, this follows from previously-shown properties. In particular, u is not in $V(T) \cap V(N)$, as this would imply $\phi(u) = u$ (because ϕ is an embedding function) and thus $\iota(u) = \iota(\phi(u))$. Then $u \notin V(N)$, which implies $\iota(u) \notin V(N_{\text{IN}})$ as ι is an isolabelling. Thus we may assume $\iota(u)$ is an internal vertex of T_{IN}, which has out-degree 2 as T_{IN} is binary. As we have just shown that u with $\iota(u) \in L \cup R \cup S$ has the same in- and out-degree in $D(N,T)$ as $\iota(u)$ in $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, this implies that u has out-degree 2 in $D(N,T)$, as required.

  For the case that $\iota(u) = \text{FUTURE}$, we observe that by construction, $\iota(u) = \iota_\mu(u)$. Furthermore $\iota_\mu(\phi_\mu(u)) \neq \text{FUTURE}$ (as $\phi_\mu(u) \in V'_L \cup V'_R \cup V_S$ and so $\phi(u) = \phi_\mu(u)$, and $\iota_\mu(\phi_\mu(u)) = \text{FUTURE}$ would imply $\iota(\phi(u)) = \iota_\mu(\phi_\mu(u)) = \text{FUTURE} = \iota(\phi(u))$, a contradiction). Thus as μ is a containment structure, we also have that u has out-degree 2 in $D(N_\mu, T_\mu)$ and so u has out-degree 2 in $D(N,T)$ as well.

  The above conditions show that ψ is a $(L \cup R \cup S, \{\text{FUTURE}\})$-containment structure, i.e. an F-partial solution. Next we show that ψ is well-behaved:

- **$D(N,T)$ contains no redundant arcs or vertices:** Suppose for a contradiction that there is a redundant arc or vertex w.r.t. ψ. We will show that such an arc or vertex is also redundant w.r.t μ, a contradiction as μ is well-behaved.

  Consider the case where a tree arc uv is redundant w.r.t ψ. Then uv is necessarily FUTURE-redundant, which implies that $\iota(u) = \iota(v) = \text{FUTURE}$. It follows by construction of ι that u, v are vertices of $D(N_\mu, T_\mu)$ with $\iota_\mu(u) = \iota_\mu(v) = \text{FUTURE}$. In addition, we have that $\iota(z) = \text{FUTURE}$ for any $z \in V(\phi(uv))$, which again implies $\iota_\mu(v) = \text{FUTURE}$ for any such edge. Finally $\phi(uv) = \phi_\psi(uv)$ (as the arc uv is in one of A'_L, A'_R, A_S) , and so $\iota_\mu(z) = \text{FUTURE}$ for all $z \in \{u, v, \} \cup V(\phi_\mu(uv))$. It follows that uv is FUTURE-redundant w.r.t μ, the desired contradiction.

  Essentially the same arguments can also be used for redundant network arcs and redundant tree or network vertices.

- **For any $y, y' \in \{\text{FUTURE}\}$ with $y \neq y$, there is no arc uv in $D(N,T)$ for which $\iota(u) = y$ and $\iota(v) = y'$:** This follows immediately from the fact that $|\{\text{FUTURE}\}| = 1$, so there are no such y, y'.

- **For any** u, v, **in** $V(D(N,T))$ **with** $\iota(v), \iota(v) \in L \cup R \cup S$**, if** $D(N,T)$ **has a path from** u **to** v **then** $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ **has a path from** $\iota(u)$ **to** $\iota(v)$**:** This has already been shown, at the start of the proof for this claim. ⌟

We now have that ψ satisfies all the conditions of a well-behaved F-partial solution. Finally we need to show that μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ.

Claim 4. *μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ.*

Proof: Let $\mu' = (D(N_{\mu'}, T_{\mu'}), \phi_{\mu'}, \iota_{\mu'})$ denote the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ. Then our aim is to show that $\mu = \mu'$.

- $D(N_\mu, T_\mu) = D(N_{\mu'}, T_{\mu'})$**:** To show this, it is enough to show that that $D(N_{\mu'}, T_{\mu'})$ is $D(N,T)$ with the arcs of $A_L \cup A_R$ and vertices of $V_L \cup V_R$ removed. To do this, we will show that after applying $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$, the LEFT-redundant arcs (resp. vertices) are exactly A_L (V_L) and the RIGHT-redundant arcs (vertices) are exactly A_R (V_R).

  Recall that for any arc or vertex a in $D(N,T)$, a is y-redundant w.r.t $(D(N,T), \phi, \iota')$ if and only if $\iota(Q_a) = \{y\}$, for some set Q_a of vertices that depends only on $D(N,T)$ and ϕ. Note that Q_a always contains a itself, if a is a vertex, or the vertices of a, if a is an arc

  So now let Q_a be the set of vertices that determines whether a is y-redundant for $(D(N,T), \phi)$, and let Q'_a denote the set of vertices that determine whether a is y-redundant for $(D(N_{\psi_L}, T_{\psi_L}), \phi_{\psi_L})$, where a is any arc or vertex that in $D(N_{\psi_L}, T_{\psi_L})$. Note that $Q'_a \subseteq Q_a$. We will show first that $\iota(Q_a) \subseteq L$ if and only if $\iota_{\psi_L}(Q'_a) \subseteq L$.

  Indeed, since $\iota(u) \in L$ implies $\iota_{\psi_L}(u) = \iota(u)$, if $\iota(Q_a) \subseteq L$ then $\iota_{\psi_L}(Q'_a) \subseteq \iota(Q_a) \subseteq L$. For the converse, if $\iota_{\psi_L}(u) \in L$ then we also have $\iota(u) = \iota_{\psi_L}(u)$, so $\iota_{\psi_L}(Q'_a) \subseteq L$ implies $\iota(Q'_a) \subseteq L$. It remains to consider the vertices of $Q_a \setminus Q'_a$. If $a = uv$ is a network arc, then $Q_a = \{u,v\} \cup Q_{a'} : a \in A(\phi(a')$ and $Q'_a = \{u,v\} \cup Q'_{a'} : a \in A(\phi_{\psi_L}(a'))$. Then $Q_a = Q'_a$, unless there is an arc a' in $D(N,T)$ with $a \in A(\phi(a'))$ and a' not in $D(N_{\psi_L}, T_{\psi_L})$. Note however that this requires that $a' \in A'_R \cup A_R$, which as argued previously would imply that $A(\phi(a')) \subseteq A'_R \cup A_R$, a contradiction to the fact that $\iota(u), \iota(v) \in L$. If a is a vertex, a similar argument applies: with $Q_a = Q'_a$ unless $D(N,T)$ has arcs in $A_R \cup A'_R$ incident to a or a' (with a' a possible vertex such that $\phi(a) = a'$ or $\phi(a') = a$). But as $\iota(a), \iota(a') \in L$, this cannot happen. Thus in either case we have $Q'_a = Q_a \subseteq L$.

  We can now show that an arc or vertex a in $D(N,T)$ is LEFT-redundant w.r.t $(D(N,T), \phi, \iota \circ g)$ if and only if a is in $D(N_{\psi_L}, T_{\psi_L})$ and a is PAST-redundant w.r.t. $(D(N_{\psi_L}, T_{\psi_L}), \phi_{\psi_L}, \iota_{\psi_L} \circ g')$, where g is the function $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$, g' is the function $\{L \to \text{PAST}\}$. Indeed we may assume that all vertices in a are mapped to elements of L, as otherwise neither side holds. Then it remains to observe that a is LEFT-redundant w.r.t $(D(N,T), \phi, \iota \circ g)$ if and only if $g(\iota(Q_a)) = \{\text{LEFT}\} \Leftrightarrow \iota(Q_a) \subseteq L \Leftrightarrow \iota_{\psi_L}(Q'_a) \subseteq L \Leftrightarrow g'(\iota_{\psi_L}(Q'_a)) = \text{PAST} \Leftrightarrow a$ is PAST-redundant w.r.t. $(D(N_{\psi_L}, T_{\psi_L}), \phi_{\psi_L}, \iota_{\psi_L} \circ g')$.

  As such, after applying $g = \{L \to \text{LEFT}, R \to \text{RIGHT}\}$ the LEFT-redundant arcs and vertices in $D(N,T)$ are exactly A_L, V_L. A similar argument shows that the RIGHT-redundant arcs and vertices are A_R, V_R. It follows that $D(N_{\mu'}, T_{\mu'})$ is exactly $D(N,T)$ with A_L, V_L, A_R, V_R removed, that is, $D(N_{\mu'}, T_{\mu'}) = D(N_\mu, T_\mu)$.

- $\phi_\mu = \phi_{\mu'}$**:** Here we use the fact that $D(N_\mu, T_\mu) = D(N_{\mu'}, T_{\mu'})$. Thus every tree arc uv of $D(N_{\mu'}, T_{\mu'})$ is in $A'_L \cup A'_R \cup A_S$, from which it follows by construction that $\phi(uv) = \phi_\mu(uv)$. Similarly as tree every vertex u of $D(N_{\mu'}, T_{\mu'})$ is in $V'_L \cup V'_R \cup V_S$, we have $\phi(u) = \phi_\mu(u)$. Then as $\phi_{\mu'}$ is the restriction of μ to the tree vertices of $V'_L \cup V'_R \cup V_S$ and the tree arcs of $A'_L \cup A'_R \cup A_S$, we have $\phi_{\mu'} = \phi_\mu$, as required.

- $\iota_\mu = \iota_{\mu'}$**:** Consider any vertex u in $D(N_\mu, T_\mu) = D(N_\mu, T_\mu)$. If $\iota_\mu(u) = \text{LEFT}$, then by construction $\iota(u) = \iota_{\psi_L}(u)$, which is in L (as $\iota_{\sigma_L}(u) = \text{PAST}$). Then by construction of μ', $\iota_{\mu'}(u) = \text{LEFT}$ as well. A similar argument shows that $\iota_{\mu'}(u) = \text{RIGHT}$ if $\iota_\mu(u) = \text{RIGHT}$. Finally if $\iota_\mu(u) \in S \cup \text{FUTURE}$, then by construction $\iota(u) = \iota_\mu(u)$, and since $\iota(u) \notin L \cup R$, we have $\iota_{\mu'}(u) = \iota(u) = \iota_\mu(u)$. Thus in all case $\iota_{\mu'}(u) = \iota_\mu(u)$, and so $\mu = \mu$, as required. ⌟

As we have now shown that ψ is a well-behaved F-partial solution and μ is the $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ, we have that μ is valid, as required. □

From the three previous lemmas we immediately have Lemma 12.

3.8 Compressing signatures

So far, we have shown relations between valid signatures such that the validity of any signature σ for a bag (P, S, F) in the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ is determined by the validity of all signatures for the child bags of (P, S, F). The final step is to contract certain long paths that may occur in the signature. This is necessary in order to bound the size, and thus the number of possible signatures, for a given bag. This is summarized by the notion of *compact* signatures, described below.

All our results relating the validity of well-behaved signatures also hold for compact signatures, that is, whether a compact signature is compact-valid can be determined by looking at the compact signatures for the child bags. (See Lemmas 23 to 26.)

Definition 11 (compact form)**.** *Let $\psi = (D(N, T), \phi, \iota)$ be a well-behaved $(S, \mathcal{Y})$-containment structure. Then the* compact form of ψ*, denoted $c(\psi)$, is the $(S, \mathcal{Y})$-containment signature derived from ψ as follows:*
For $y \in \mathcal{Y}$:

- *if there is a path x_1, x_2, x_3 in N with x_2 having in-degree 1 and out-degree 1 in N, and $\iota(\{x_1, x_2, x_3\}) = \{y\}$, and $\phi(u) \neq x_2$ for any vertex u in T, then delete vertex x_2 and arcs x_1x_2, x_2x_3 from N, and add the arc x_1x_3. For the arc uv in T for which $x_1x_2x_3$ is part of the path $\phi(uv)$, replace $x_1x_2x_3$ in $\phi(uv)$ with x_1x_3.*

That is, we suppress a vertex x in N (and adjust ϕ as necessary), if x has a single in-neighbour and out-neighbour, and all three are mapped to a label $y \in Y$ by ι, and no vertex of T is mapped to x by ϕ.

We call such a path a long y-path *and refer to the above process as* suppressing long y-paths*.*

If $c(\psi) = \psi$ then we say ψ is a compact $(S, \mathcal{Y})$*-containment structure.*

We call a compact signature σ for (P, S, F) compact-valid *if there is a compact F-partial solution ψ such that $\sigma = c(\sigma')$ for σ' is the $(P \to \text{PAST})$-restriction of ψ.*

Note that by definition, any compact $(S, \mathcal{Y})$-containment structure is also a well-behaved $(S, \mathcal{Y})$-containment structure.

Lemma 17. *If ψ is a well-behaved $(S, \mathcal{Y})$-containment structure, then $c(\psi)$ is a compact $(S, \mathcal{Y})$-containment structure.*

Proof: Let $\psi = (D(N,T), \phi, \iota)$ and let $c(\psi) = (D'(N',T'), \phi', \iota')$. It is clear from the construction of $c(\psi)$ that $D'(N',T')$ is a display graph and that ϕ' is an embedding of T' into N'. Moreover $\iota'(u) = \iota(u)$ for each vertex u in $D'(N',T')$, and the only vertices of $D(N,T)$ that were suppressed in the construction of $D'(N,T')'$ were vertices v for which $\iota(v) \in \{\text{PAST}, \text{FUTURE}\}$. As such ι' satisfies all the requirements for the isolabelling function in a $(S,\mathcal{Y})$-containment structure, and so $(D'(N',T'), \phi', \iota')$ is a $(S,\mathcal{Y})$-containment structure.

It is also clear that $c(\psi)$ is well-behaved, by construction and the fact that ψ is well-behaved.

Finally observe that $D(N',T')$ contains no long y-paths for any $y \in \mathcal{Y}$, a all such paths are suppressed in the construction of ψ'. □

We will now show that for the purposes of our dynamic programming algorithm, it is enough to restrict our attention to compact signatures. This allows us to get an FPT bound on the number of signatures we have to consider. In order to do this, we define an analogue of "restriction" that allows us to derive compact $(S',\mathcal{Y})$-containment structures from $(S,\mathcal{Y})$-containment structures.

Definition 12 (compact restriction)**.** *Let $\psi = (D(N,T), \phi, \iota)$ be an $(S,\mathcal{Y})$-containment structure, let $S' \subseteq S$, and let $g : S \cup \mathcal{Y} \to S' \cup \mathcal{Y}$ be a restriction function (i.e. such that for all $v \in S \cup \mathcal{Y}$, $g(v) = v$ if $v \in S'$ and $g(v) \in \mathcal{Y}$ otherwise). Then we define the* compact-g-restriction *of ψ to be the compact form of the g-restriction of ψ.*

When σ is a compact signature for (P,S,F), we say that σ is compact-valid *if there is a compact F-partial solution ψ such that σ is the compact-$\{P \to \text{PAST}\}$-restriction of ψ.*

Similarly, when μ is a compact reconciliation for $(L \cup R, S, F)$, we say μ is compact-valid *if there is a compact F-partial solution ψ such that μ is the compact-$\{L \to \text{LEFT}, R \to \text{RIGHT}\}$-restriction of ψ.*

Definition 13. *Let $\sigma = (D(N,T), \phi, \iota)$ and $\sigma_0 = (D(N_0,T_0), \phi_0, \iota_0)$ be $(S,\mathcal{Y})$-containment structures. We say σ_0 is a* subdivision *of σ if σ_0 can be derived from σ as follows:*

- *For each network arc uv in $D(N,T)$ with $\iota(u) = \iota(v) = y \in Y$, replace uv with a path $u_1 = u, u_2, \ldots, u_j = v$ for some $j \geq 2$, where u_i is a new vertex for each $1 < i < j$.*
- *For any new path $u_1 = u, \ldots, u_j = v$ created this way, if uv is part of the path $\phi(u'v')$ for a tree arc $u'v'$ then replace uv in this path with $u_1, \ldots u_j$.*
- *For any new path $u_1 = u, \ldots, u_j = v$ created this way, set $\iota(u_i) = \iota(u)$ for each $1 < i < j$.*

Observe that $\sigma = c(\sigma_0)$ if and only if σ is compact and σ_0 is a subdivision of σ.

Observe also that for any σ, σ_0 such that σ_0 is a subdivision of σ, $c(\sigma) = c(\sigma_0)$.

Lemma 18. *Let $\sigma_0 = (D(N_0,T_0), \phi_0, \iota_0)$ be a subdivision of $\sigma = (D(N,T), \phi, \iota)$, and for each arc uv in $A(D(N,T))$ let P_{uv} denote the set of arcs in $D(N_0,T_0)$ on the path from u to v corresponding to the subdivision of uv. (Taking $P_{uv} = \{uv\}$ if uv was not subdivided.)*

Let $\sigma_0' = (D(N_0',T_0'), \phi_0', \iota_0')$ be the g-restriction of σ_0 and let $\sigma' = (D(N',T'), \phi', \iota')$ be the g-restriction of σ, for some restriction function g.

Then $A(D(N_0',T_0')) = \bigcup_{uv \in A(D(N',T'))} P_{uv}$. Thus, σ_0' is a subdivision of σ'.

Proof: It is sufficient to show that for each $a \in A(D(N,T))$, any arc or internal vertex of P_a in σ_0 is y-redundant w.r.t. $(D(N_0,T_0), \phi_0, \iota_0 \circ g)$ if and only if a is y-redundant w.r.t. $(D(N,T), \phi, \iota \circ g)$

This can be seen by construction of σ_0, in particular the fact that P_a is part of the path $\phi_0(u'v')$ if and only if a is part of the path $\phi(u'v')$, and the fact that $\iota_0(z) = \iota(u) = \iota(v)$ (and thus $g(\iota(u))) = g(\iota(u)) = g(\iota(v)))$ for every vertex in P_a. □

Lemma 19. *Let σ_0 be a $(S, \mathcal{Y})$-containment structure, let $\sigma = c(\sigma_0)$, and let σ'_0 be the g-restriction of σ_0 for some restriction function $g : S \cup \mathcal{Y} \to S' \cup \mathcal{Y}$. Then $c(\sigma'_0)$ is the compact-g-restriction of σ.*

Proof: Let σ' be the g-restriction of σ. As σ_0 is a subdivision of σ and σ'_0 is the g-restriction of σ_0, Lemma 18 implies that σ'_0 is a subdivision of σ'. It follows that $c(\sigma'_0) = c(\sigma')$, which is the compact-$g$-restriction of σ by construction. □

Lemma 20. *Let σ be a well-behaved signature for a bag (P, S, F). If σ is valid then $c(\sigma)$ is compact-valid.*

Similarly if μ is a well-behaved reconciliation for a Join bag $(L \cup R, S, F)$ and μ is valid, then $c(\mu)$ is compact-valid.

Proof: Suppose a signature σ is valid and let ψ be the F-partial solution such that σ is the $\{P \to \text{PAST}\}$-restriction of ψ. Then $c(\psi)$ is compact F-partial solution, and by Lemma 19, $c(\sigma)$ is the compact-$\{P \to \text{PAST}\}$-restriction of $c(\psi)$. Thus $c(\psi)$ is compact-valid, as required.

For the case that a reconciliation μ is valid, a similar argument holds, using the restriction function $\{L \to \text{LEFT}, R \to \text{RIGHT}\}$ instead of $\{P \to \text{PAST}\}$. □

Lemma 21. *Let σ be a compact signature for a bag (P, S, F). If σ is compact-valid then there is a valid signature σ_0 for (P, S, F) such that $\sigma = c(\sigma_0)$ is compact-valid.*

Similarly if μ is a compact reconciliation for a Join bag $(L \cup R, S, F)$ and μ is compact-valid, then there is a valid reconciliation μ_0 for$(L \cup R, S, F)$ such that $\mu = c(\mu_0)$.

Proof: Suppose a compact signature σ is compact-valid and let ψ be the compact F-partial solution such that σ is the compact-$\{P \to \text{PAST}\}$-restriction of ψ. Let σ_0 be the $\{P \to \text{PAST}\}$-restriction of ψ. Then σ_0 is a valid signature for (P, S, F), and by construction $\sigma = c(\sigma_0)$.

For the case that a compact reconciliation μ is compact-valid, a similar argument holds. □

Lemma 22. *Let σ' be the $(S_1 \to y_1, \ldots, S_j \to y_j)$-restriction of some $(S, \mathcal{Y})$-containment structure σ. If σ is compact, then σ' has a long y-path only if $y \in \{y_1, \ldots, y_j\}$.*

Proof: Let $\sigma = (D(N, T), \phi, \iota)$ and $\sigma' = D(N', T'), \phi', \iota')$. Suppose for a contradiction that σ has a long y-path for some $y \notin \{y_1, \ldots, y_j\}$. Thus in particular, there is a path x_1, x_2, x_3 in N' with x_2 having in-degree and out-degree 1, and $\iota'(x_1) = \iota(x_2) = \iota'(x_3) = y$. Then by construction, x_1, x_2, x_3 is also a path in N, and $\iota(x_1) = \iota(x_2) = \iota(x_3) = y$ (since the restriction function $(S_1 \to y_1, \ldots, S_j \to y_j)$ does not assign y to any vertex that was not already labelled y by ι). Furthermore x_2 cannot have any other incident arcs beside x_1x_2 and x_2x_3, as such arcs would not become redundant after applying $(S_1 \to y_1, \ldots, S_j \to y_j)$, and so such arcs would be in N' as well. It follows that σ has a long y-path, contradicting the assumption that σ is compact. □

Corollary 1. $(N_{\text{IN}}, T_{\text{IN}})$ *is a* YES*-instance of* TREE CONTAINMENT *if and only if there is a compact-valid signature $\sigma = (D(N, T), \phi, \iota)$ for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\iota^{-1}(\text{FUTURE}) = \varnothing$.*

Proof: By Lemma 8, we have that $(N_{\text{IN}}, T_{\text{IN}})$ is a YES-instance of TREE CONTAINMENT if and only if there is a valid signature $\sigma_0 = (D(N_0, T_0), \phi_0, \iota_0)$ for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with ${\iota_0}^{-1}(\text{FUTURE}) = \varnothing$.

So suppose that such a signature σ_0 exists. Let $\sigma = c(\sigma_0)$, and observe that σ also satisfies $\iota^{-1}(\text{FUTURE}) = \varnothing$. Furthermore by Lemma 20, σ is compact-valid.

Conversely suppose there is a compact-valid signature $\sigma = (D(N, T), \phi, \iota)$ for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\iota^{-1}(\text{FUTURE}) = \varnothing$. Then by Lemma 21 there is a valid signature σ_0 for $(V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})), \varnothing, \varnothing)$ with $\sigma = c(\sigma_0)$. Then as σ_0 is a subdivision of σ, we also have ${\iota_0}^{-1}(\text{FUTURE}) = \varnothing$. Thus by Lemma 8, $(N_{\text{IN}}, T_{\text{IN}})$ is a YES-instance of TREE CONTAINMENT. □

We are now ready to prove the compact equivalents of the main lemmas for Forget, Introduce and Join bags

Lemma 23 (Leaf bag)**.** *Let (P, S, F) correspond to a Leaf bag in the tree decomposition i.e. $P = S = \varnothing$, $F = V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$ and (P, S, F) has no children. Then a compact signature $\sigma = (D(N, T), \phi, \iota)$ for (P, S, F) is compact-valid if and only if $\iota^{-1}(\text{PAST}) = \varnothing$.*

Proof: Similar to the proof of Lemma 9, we have that if $\psi = (D(N', T'), \phi', \iota')$ is a compact F-partial solution and σ is the compact-$\{P \to \text{PAST}\}$-restriction of ψ, then because $P = \varnothing$ we must have $\iota(u) \neq \text{PAST}$ for all $u \in V(D(N, T))$. Conversely if $\iota^{-1}(\text{PAST}) = \varnothing$ then σ is a compact F-partial solution and also the compact-$\{P \to \text{PAST}\}$-restriction of itself. □

Lemma 24 (Forget bag)**.** *Let (P, S, F) correspond to a Forget bag in the tree decomposition with child bag (P', S', F'), i.e. $P = P' \cup \{z\}$, $S = S' \setminus \{z\}$ and $F = F'$.*

Then a compact signature σ for (P, S, F) is compact-valid if and only if there is a compact-valid signature σ' for (P', S', F') such that σ is the compact-$\{\{z\} \to \text{PAST}\}$-restriction of σ'.

c.f. Lemma 10

Proof: Suppose first that σ is compact-valid. Then exists a valid signature σ_0 for (P, S, F) with $c(\sigma_0) = \sigma$ (Lemma 21). Then by Lemma 10, there is a valid signature σ_0' for (P', S', F') such that σ_0 is the $\{\{z\} \to \text{PAST}\}$-restriction of σ_0'. Then let $\sigma' = c(\sigma_0')$ and observe that σ' is compact-valid (Lemma 20). Furthermore by Lemma 19, σ is the compact-$\{\{z\} \to \text{PAST}\}$-restriction of σ', as required.

Conversely, suppose there is a compact-valid signature σ' for (P', S', F') such that σ is the compact-$\{\{z\} \to \text{PAST}\}$-restriction of σ'. Then there is a valid signature σ_0' for (P', S', F') with $\sigma' = c(\sigma_0')$ (Lemma 21). Let σ_0 be the $\{\{z\} \to \text{PAST}\}$-restriction of σ_0'. Then by Lemma 10, σ_0 is valid. Furthermore by Lemma 19, $c(\sigma_0)$ is the compact-$\{\{z\} \to \text{PAST}\}$-restriction of $c(\sigma_0') = \sigma'$. That is $c(\sigma_0) = \sigma$. Then as σ_0 is valid, σ is compact-valid (Lemma 20). □

Lemma 25. *Let (P, S, F) correspond to an Introduce bag in the tree decomposition with child bag (P', S', F'), i.e. $P' = P$, $S' = S \setminus \{z\}$ and $F' = F \cup \{z\}$.*

Then a compact signature σ for (P, S, F) is valid if and only if σ' is a compact-valid signature for (P', S', F'), where σ' is the compact-$\{\{z\} \to \text{FUTURE}\}$-restriction of σ.

c.f. Lemma 11

Proof: Suppose first that σ is compact-valid. Then there is a valid signature σ_0 for (P, S, F) such that $c(\sigma_0) = \sigma$ (Lemma 21). Then let σ_0' be the $\{\{z\} \to \text{FUTURE}\}$-restriction of σ_0, and observe that σ_0' is valid by Lemma 11. Then $c(\sigma_0')$ is compact-valid (Lemma 20). Furthermore by Lemma 19, $c(\sigma_0')$ is

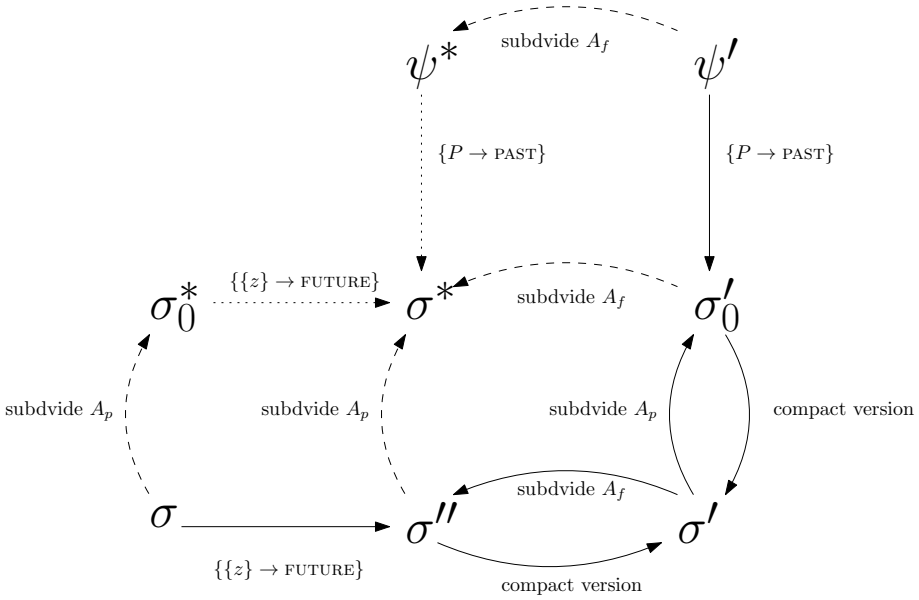

Figure 9: Illustration of proof for Introduce bags, compact-validity of σ' implies compact-validity of σ'. Solid lines are relations that we may assume; the dashed line shows the construction of σ^*, ψ^* and σ'_0; dotted lines are restriction relations we can show using Lemma 18.

the compact-$\{\{z\} \to \text{FUTURE}\}$-restriction of $c(\sigma_0) = \sigma$. Then letting $\sigma' = c(\sigma'_0)$ we have that σ' is compact-valid and the compact-$\{\{z\} \to \text{FUTURE}\}$-restriction restriction of σ, as required.

For the converse, assume $\sigma' = (D(N', T), \phi', \iota')$ is compact-valid, and let ψ' be the compact F'-partial solution such that σ' is the compact-$\{P \to \text{PAST}\}$-restriction of ψ'. Let σ'_0 be the $\{P' \to \text{PAST}\}$-restriction of ψ', so σ'_0 is a subdivision of σ'. In addition let σ'' be the $\{\{z\} \to \text{FUTURE}\}$-restriction of σ, so σ'' is also a subdivision of σ'. Let A_p be the subset of arcs in N' that are subdivided by one or more additional vertices to produce σ'_0. Similarly let A_f be the subset of arcs in N' that are subdivided by one or more additional vertices to produce σ''.

By Lemma 22, σ'' has no long PAST-paths and σ'_0 has no long FUTURE-paths. Thus the sets of arcs A_p and A_f are disjoint (as $\iota'(u) = \iota'(v) = \text{PAST}$ for any $uv \in A_p$, and $\iota'(u) = \iota'(v) = \text{FUTURE}$ for any $uv \in A_f$). So now for each arc uv in N', define P_{uv} to be the path corresponding to uv in σ'_0 if $uv \in A_p$, let P_{uv} be the path corresponding to uv in σ'' if $uv \in A_f$, and let P_{uv} be the single arc uv otherwise.

Now define σ^* to be the subdivision of σ' derived by replacing every arc uv in N by P_{uv}. Now we have that σ^* is also a subdivision of σ'_0 (by subdividing arcs of A_p) and also of σ'' (by subdividing arcs of A_f, and also that $\sigma' = c(\sigma^*)$ (as σ' is compact). (See Fig. 9.)

We now claim that σ^* is a valid signature for (P', S', F'). To see this, Let ψ^* be the subdivision of ψ' derived by replacing uv with P_{uv} for any $uv \in A_f$ (observe that all arcs of A_f appear in the display graph of ψ, as they are all arcs in σ' and were not subdivided to create σ'_0). Then ψ^* is also a well-behaved F'-partial solution. Then using Lemma 18 and the fact that σ'_0 is the $\{P' \to \text{PAST}\}$-restriction of ψ', we have that the $\{P' \to \text{PAST}\}$-restriction ψ^* can be derived from σ'_0 by replacing the uv with P_{uv} for all

$uv \in A_f$. That is, the $\{P' \to \text{PAST}\}$-restriction ψ^* is exactly σ^*. Thus σ^* is indeed valid.

Next, let σ_0^* be the subdivision of σ derived by replacing uv with P_{uv} for any $uv \in A_p$. Similar to the case with ψ^*, we can show using Lemma 18 that the $\{\{z\} \to \text{FUTURE}\}$-restriction of σ_0^* is σ'' with uv replaced by P_{uv} for all $uv \in A_p$, i.e. σ^*. So we now have that σ^* is the $\{\{z\} \to \text{FUTURE}\}$-restriction of σ_0^* and also a valid signature for (P', S', F'). By Lemma 11, this implies that σ_0^* is valid. Then σ, which is the compact version of σ_0^*, is compact-valid by Lemma 20. □

Lemma 26. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$ and let σ be a compact signature for $(L \cup R, S, F)$. Then, σ is compact-valid if and only if there is a compact-valid reconciliation μ for $(L \cup R, S, F)$ such that σ is the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ.*

c.f. Lemma 14.

Proof: Suppose first that σ is compact-valid. Then there is a valid signature σ_0 for $(L \cup R, S, F)$ such that $\sigma = c(\sigma_0)$ (Lemma 21.) By Lemma 14, there is a valid reconciliation μ_0 for $(L \cup R, S, F)$ such that σ_0 is the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ_0. Then let $\mu = c(\mu_0)$, so μ is a compact-valid reconciliation (Lemma 20). By Lemma 19, the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of $\mu = c(\mu_0)$ is $c(\sigma_0) = \sigma$, as required.

Conversely, suppose that μ is a compact reconciliation for $(L \cup R, S, F)$ such that σ is the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ. As μ is compact-valid, there is a valid reconciliation μ_0 for $(L \cup R, S, F)$ such that $\mu = c(\mu_0)$ (Lemma 21). Let σ_0 be the $\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ. Then by Lemma 14, σ_0 is a valid signature for $(L \cup R, S, F)$. Moreover by Lemma 19, $c(\sigma_0)$ is the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of $\mu = c(\mu_0)$, i.e. $c(\sigma_0) = \sigma$. Then as σ_0 is valid, σ is compact-valid (Lemma 20). □

Lemma 27. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let μ be a compact reconciliation for $(L \cup R, S, F)$. Let σ_L be the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and σ_R the compact-$\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ. If μ is compact-valid, then σ_L is a compact-valid signature for $(L, S, F \cup R)$ and σ_R is a compact-valid signature for $(R, S, F \cup L)$.*

c.f. Lemma 15

Proof: Suppose μ is compact-valid. Then there is a valid reconciliation μ_0 for $(L \cup R, S, F)$ such that $\mu = c(\mu_0)$ (Lemma 21). Now let σ'_L be the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ_0. Then by Lemma 15, σ'_L is a valid signature for $(L, S, F \cup R)$. Furthermore by Lemma 19, $c(\sigma'_L)$ is the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of $c(\mu_0) = \mu$. That is, $c(\sigma_L) = \sigma_L$. It follows by Lemma 20 that σ_L is compact-valid, as required. As similar argument shows that σ_R is a compact-valid signature for $(R, S, F \cup L)$. □

Lemma 28. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let μ be a compact reconciliation for $(L \cup R, S, F)$. Let σ_L be the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and σ_R the compact-$\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ. If σ_L is a compact-valid signature for $(L, S, F \cup R)$ and σ_R is a compact-valid signature for $(R, S, F \cup L)$, then μ is compact-valid.*

c.f. Lemma 16

Proof: In what follows, let $\sigma = (D(N_\sigma, T_\sigma), \phi_\sigma, \iota_\sigma)$, for any given containment structure σ.

The proof is along similar lines to Lemma 25, but with more containment structures. Before we can proceed, we first show that the the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ is in fact compact, and therefore σ_L is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, as well as the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ.

Indeed, let σ''_L denote the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and suppose for a contradiction that σ''_L contains a long y-path for some $y \in \{\text{PAST}, \text{FUTURE}\}$. Thus there is a path x_1, x_2, x_3 in $N_{\sigma''_L}$ where x_2 has in-degree and out-degree 1 and $\iota_{\sigma''_L}(x_1) = \iota_{\sigma''_L}(x_2) = \iota_{\sigma''_L}(x_3) = y$. If $y = \text{PAST}$ then $\iota_\mu(x_1) = \iota_\mu(x_2) = \iota_\mu(x_3) = \text{LEFT}$. Otherwise $y = \text{FUTURE}$, and $\iota_\mu(x_1), \iota_\mu(x_2), \iota_\mu(x_3)$ are all in $\{\text{RIGHT}, \text{FUTURE}\}$. But note that all three of $\iota_\mu(x_1), \iota_\mu(x_2), \iota_\mu(x_3)$ must be the same value, otherwise μ has an arc uv with $\iota_\mu(\{u, v\}) = \{\text{RIGHT}, \text{FUTURE}\}$ and μ is not well-behaved. So if x_2 has in-degree and out-degree 1 in N_μ, then μ has a long y-path for $y = \iota_\mu(u)$, a contradiction as μ is compact. Otherwise, suppose x_2 has an incident arc x_2z for $z \neq x_3$ (the case of an incident arc zx_2 is similar). As x_2z is not an arc in $N_{\sigma''_L}$, it must hold that $\iota_\mu(z) = \iota_\mu(x_2)$ (otherwise the arc would not become redundant or μ is not well-behaved). Then since x_2z is not redundant w.r.t μ, there is some tree arc uv in T_μ such that x_2z is an arc of $\phi_\mu(uv)$. But this implies also that x_1x_2 is also arc of $\phi_\mu(uv)$, as $x_2 \neq \iota_\mu(u)$. Thus x_2z is deleted in the construction of σ''_L only if uv is, which would in turn imply that x_1x_2 is deleted, again a contradiction.

So we may assume that σ''_L is compact and thus $\sigma''_L = c(\sigma''_L) = \sigma_L$, so σ_L is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ. A similar argument shows that σ_R is the $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ.

So now assume that σ_L and σ_R are both compact-valid. Let ψ_L denote the compact $F \cup R$-partial solution for which σ_L is the compact $\{L \to \text{PAST}\}$-restriction, and let σ'_L be the $\{L \to \text{PAST}\}$-restriction of ψ_L, so that $\sigma_L = c(\sigma'_L)$. Similarly let ψ_R denote the compact $F \cup L$-partial solution for which σ_R is the compact $\{R \to \text{PAST}\}$-restriction, and let σ'_R be the $\{R \to \text{PAST}\}$-restriction of ψ_R, so that $\sigma_R = c(\sigma'_R)$. (See Fig. 10)

By Lemma 22, the only long-y-paths in σ'_L are for $y = \text{PAST}$. So σ'_L is a subdivision of σ_L in which the only subdivided arc are those uv for which $\iota_{\sigma_L}(u) = \iota_{\sigma_L}(v) = \text{PAST}$. So let A_L the set of these arcs, and for each $uv \in A_L$ let P_{uv} be the corresponding path in $N_{\sigma'_L}$. Similarly let A_R be the set of arcs in N_{σ_R} that are subdivided to produce σ'_R from σ_R (noting that $\iota_{\sigma_R}(u) = \iota_{\sigma_R}(v) = \text{PAST}$ for any $uv \in A_R$), and let P_{uv} be the corresponding path in $N_{\sigma'_R}$ for each $uv \in A_R$. Observe that since $\iota_{\sigma_L}(u) = \text{PAST}$ implies $\iota_\mu(u) = \text{LEFT}$ and $\iota_{\sigma_R}(u) = \text{PAST}$ implies $\iota_\mu(u) = \text{RIGHT}$, the arc sets A_L and A_R are disjoint.

Now let σ^*_L be the subdivision of σ'_L (not σ_L) by replacing uv with P_{uv} for all $uv \in A_R \cap A(N_{\sigma'_L})$. Equivalently, σ^*_L is the subdivision of σ_L derived by replacing uv with P_{uv} for all arcs uv in $A_L \cup (A_R \cap A(N_{\sigma_L}))$. Thus $c(\sigma^*_L) = \sigma_L$. Define σ^*_R analogously.

We claim that σ^*_L is a valid signature for $(L, S, R \cup F)$. To see this, let ψ^*_L be the subdivision of ψ_L derived by replacing uv with P_{uv} for every uv in $A_L \cup (A_R \cap A(N_{\psi_L}))$. Then by Lemma 18 and the fact that σ'_L is the $\{L \to \text{PAST}\}$-restriction of ψ_L, we have that σ^*_L is the $\{L \to \text{PAST}\}$-restriction of ψ^*_L. Thus ψ^*_L is valid, and a similar argument shows that ψ^*_R is valid.

Now let μ^* (respectively μ^*_L, μ^*_R) be the subdivision of μ derived by replacing uv with P_{uv} for all uv in $A_L \cup A_R$ (respectively A_L, A_R). Note that μ^* is also a subdivision of both μ^*_L and μ^*_R.

Using a similar approach to before, by Lemma 18 and the fact that σ_L is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, we can show that σ'_L is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ^*_L. Using this fact in turn together with Lemma 18, we can show that σ^*_L is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ^*. Similarly we can show that σ^*_R is the $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-

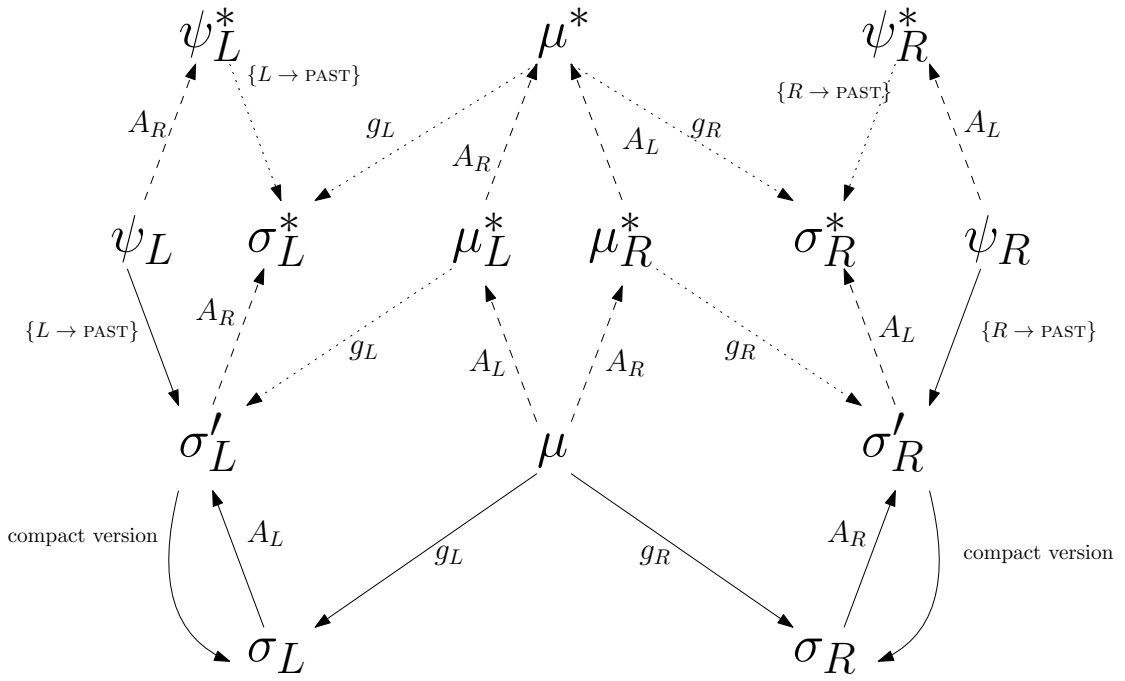

Figure 10: Illustration of proof of Lemma 28. Solid lines are relations that we may assume; the dashed lines show the construction of $\sigma_L^*, \sigma_R^*, \psi_L^*, \psi_R^*, \mu_L^*, \mu_R^*, \mu^*$; dotted lines are restriction relations we can show using Lemma 18. Here g_L denotes the restriction function $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$ and g_R denotes $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$; labels A_L or A_R indicate a subdivision of those arcs

restriction of μ^*.

If σ_L^* is a valid signature for $(L, S, F \cup R)$ which is the $\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ^*, and σ_R^* is a valid signature for $(R, S, F \cup L)$ which is the $\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ^*, we can now apply Lemma 14 to see that μ^* is a valid reconciliation for $(L \cup R, S, F)$.

It remains to observe that as μ^* is valid and $\mu = c(\mu^*)$, Lemma 20 implies that μ^* is compact-valid. □

From Lemmas 26 to 28 we have the following:

Corollary 2. *Let $(L \cup R, S, F)$ be a Join bag with child bags $(L, S, F \cup R)$ and $(R, S, F \cup L)$, and let σ be a compact signature for $(L \cup R, S, F)$. Then σ is compact-valid if and only if there is a compact reconciliation μ for $(L \cup R, S, F)$, and compact-valid signatures σ_L for $(L, S, F \cup R)$ and σ_R for $(R, S, F \cup L)$, such that σ is the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ, σ_L is the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ, and σ_R is the compact-$\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ.*

4 Algorithm and running time

Algorithm 1 gives a summary of our algorithm for TREE CONTAINMENT. To summarise: we compute for each bag $x = (P, S, F)$ in the tree decomposition, a set CV_x of compact-valid signatures for x - that is,

Algorithm 1: Tree Containment ($D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$)

Compute $tw(N_{\text{IN}})$;
if $tw(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) > 2tw(N_{\text{IN}}) + 1$ **then return** FALSE;
Compute a nice minimum-width tree decomposition $\mathcal{T}$ of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$;
foreach *bag* $x = (P, S, F)$ *of* $\mathcal{T}$ *in a bottom-up traversal* **do**
    if x *is a Leaf bag* **then**
        $CV_x \leftarrow$ set of all compact signatures $\sigma = (D(N, T), \phi, \iota)$ for x with $\iota^{-1}(\text{PAST}) = \varnothing$;
    else if x *is a Forget bag with child* $y = (P \setminus \{z\}, S \cup \{z\}, F)$ *in* $\mathcal{T}$ **then**
        foreach $\sigma \in CV_y$ **do**
            add the compact-$\{z \to \text{PAST}\}$-restriction of σ to CV_x
    else if x *is an Introduce bag with child* $y = (P, S \setminus \{z\}, F \cup \{z\})$ *in* $\mathcal{T}$ **then**
        foreach *compact signature* σ *of* x **do**
            if CV_y *contains the compact-*$\{z \to \text{FUTURE}\}$*-restriction of* σ **then**
                add σ to CV_x;
    else if x *is a Join bag with children* $y_L = (L, S, R \cup F)$ & $y_R = (R, S, L \cup F)$ **then**
        foreach *compact reconciliation* μ *for* x **do**
            $\sigma_L \leftarrow$ the compact-$\{\text{LEFT} \to \text{PAST}, \text{RIGHT} \to \text{FUTURE}\}$-restriction of μ;
            $\sigma_R \leftarrow$ the compact-$\{\text{RIGHT} \to \text{PAST}, \text{LEFT} \to \text{FUTURE}\}$-restriction of μ;
            if $\sigma_L \in CV_{y_L}$ *and* $\sigma_R \in CV_{y_R}$ **then**
                add the compact-$\{\{\text{LEFT}, \text{RIGHT}\} \to \text{PAST}\}$-restriction of μ to CV_x;
foreach $(D(N, T), \phi, \iota) \in CV_{\text{root}(\mathcal{T})}$ **do**
    if $\iota^{-1}(\text{FUTURE}) = \varnothing$ **then return** TRUE;
return FALSE

compact signatures for which there exists a corresponding F-partial solution. The algorithm uses compact-restrictions to convert a compact signature of one bag into a compact signature for a different bag. Recall that such a restriction works by mapping certain vertices to a label PAST or FUTURE, removing redundant parts of the display graph and collapsing long paths, similarly to the process described in Section 2.3. See Sections 3.3 and 3.8 for the formal definitions. For join bags, the algorithm uses reconciliations, 3-way analogues of signatures, using labels $\{\text{LEFT}, \text{RIGHT}, \text{FUTURE}\}$ instead of $\{\text{PAST}, \text{FUTURE}\}$, see Section 3.7.

The correctness of the computation of the sets CV_x follows from Lemmas 23 to 25 and Corollary 2. The correctness of the last three lines, in which we return TRUE if and only if there is a compact-valid signature $(D(N, T), \phi, \iota)$ for the root bag with $\iota^{-1}(\text{FUTURE}) = \varnothing$, is a consequence of Corollary 1.

To show that the running time is bounded in a function in the treewidth of N, the main challenge is to bound the number of compact signatures for a bag (P, S, F) by a function of $|S|$ (which, by Theorem 1, we may assume is at most $2tw(N) + 1$). In order to do this, we first bound the size of the display graph $D(N, T)$ in a signature by a function of $|S|$. We will then use this to bound the number of possible display graphs, embedding functions and isolabellings, and thus the number of compact signatures.

Lemma 29. *Any compact signature $\sigma = (D(N,T), \phi, \iota)$ for a bag (P, S, F) satisfies $|V(D(N,T)))| \in O(|S|)$. Any compact reconciliation $\mu = (D(N,T), \phi, \iota)$ satisfies $|V(D(N,T)))| \in O(|S|)$.*

Proof: Let $\sigma = (D(N,T), \phi, \iota)$ be a compact signature for (P, S, F).

We first bound the number of arcs in $D(N,T)$. To do this, let A_S denote the subset of arcs in $D(N,T)$ incident to a vertex in $\iota^{-1}(S)$. As there is only one vertex u with $\iota(u) = s$ for each $s \in S$ and every vertex in $D(N,T)$ has total degree at most 3, we have that $|A_S| \leq 3|S|$. As $\phi(uv)$ and $\phi(u'v')$ are arc-disjoint for any distinct tree arcs $uv, u'v'$, it follows that there are at most $|A_S \cap A(N)|$ arcs uv of T for which $\phi(uv)$ contains an arc from A_S. There are at most $|A_S \cap A(T)|$ arcs in T incident to a vertex from $\iota^{-1}(S)$. Thus there are at most $|A_S|$ arcs uv in T for which $\{u, v\} \cup V(\phi(uv))$ contains a vertex from $\iota^{-1}(S)$.

Every remaining arc uv in T has $\iota(\{u, v\} \cup V(\phi(uv))) \subseteq \{\text{PAST}, \text{FUTURE}\}$. Furthermore as σ is well-behaved, we must have that $\iota(u) = \iota(v)$, and $\iota(u') = \iota(v')$ for every arc $u'v'$ in the path $\phi(uv)$. It follows that for all but at most $|A_S| \leq 3|S|$ arcs of T, we have $\iota(u) = \iota(v) \in \{\text{PAST}, \text{FUTURE}\}$ and $\iota(V(\phi(uv))) = \{\text{PAST}\}$ or $\iota(V(\phi(uv))) = \{\text{FUTURE}\}$.

If $\iota(u) = \iota(v) = \text{PAST}$ and $\iota(V(\phi(uv))) = \{\text{PAST}\}$, then uv is redundant w.r.t $\{\text{PAST}\}$, a contradiction as we may assume no valid signature has a $\{\text{PAST}\}$-redundant arc. Similarly we have a contradiction if $\iota(u) = \iota(v) = \text{FUTURE}$ and $\iota(V(\phi(uv))) = \{\text{FUTURE}\}$. So it must be the case that for all but at most $3|S|$ tree arcs uv, either $\iota(u) = \iota(v) = \text{PAST}$ and $\iota(\phi(uv)) = \{\text{FUTURE}\}$, or $\iota(u) = \iota(v) = \text{FUTURE}$ and $\iota(\phi(uv)) = \{\text{PAST}\}$. In particular, we have that $\iota(\phi(v)) \neq \iota(v)$ and, so v has out-degree 2

It follows any lowest arc uv in T is one of the at most $|A_S|$ arcs $\{u, v\} \cup V(\phi(uv))$ that contains a vertex from $\iota^{-1}(S)$. Thus in total T has at most $2|A_S| \leq 6|S|$ arcs.

To bound the arcs of N, observe that any arc $uv \in A(N)$ not in A_S satisfies $\iota(u) = \iota(v) = \text{PAST}$ or $\iota(u) = \iota(v) = \text{FUTURE}$ (it cannot be that $\{\iota(u), \iota(v)\} = \{\text{PAST}, \text{FUTURE}\}$ as σ is well-behaved). Then uv must be part of the path $\phi(u'v')$ for some tree arc $u'v'$ (otherwise uv is redundant w.r.t σ, a contradiction). So now let A'_N denote the set of arcs in N that are part of a path $\phi(uv)$ for some tree arc uv (note that A'_N and A_S are not necessarily disjoint but $A(N) \subseteq A'_N \cup A_S$).

For any internal vertex z on a path $\phi(uv)$, we must have that z is incident to an arc from A_S. Indeed suppose this is not the case, then z and its neighbours in $\phi(uv)$ form a path x_1, x_2, x_3 with $\iota(x_1) = \iota(x_2) = \iota(x_3) = y \in \{\text{PAST}, \text{FUTURE}\}$. This forms a long-$y$-path (contradicting the fact that σ is compact), unless $z = x_2$ is incident to another arc in $A(N)$. But such an arc cannot be in A_S by assumption, and also cannot be part of a path $\phi(u'v')$ for any tree arc $u'v'$ (as $\phi(uv)$ and $\phi(u'v')$ share a vertex z only if $uv, u'v'$ share a vertex w with $\phi(w) = z$). Thus there is no other arc incident to z, and we have that σ is not compact, a contradiction.

Thus we now have that every internal vertex of a a path $\phi(uv)$ is incident to an arc from A_S, and thus there are at most $2|A_S| \leq 6|S|$ such vertices. As there at most $|A(T)| \leq 2|A_S|$ paths $\phi(uv)$, we have that $|A'_N| \leq 2|A_S| + 2|A_S| \leq 12|S|$.

Thus in total, the number of arcs in $D(N,T)$ is at most $|A(T)|+|A_S|+|A'_N| \leq 2|A_S|+|A_S|+2|A_S| = 5|A_S| \leq 15|S|$. It follows that the number of non-isolated vertices in $D(N,T)$ is at most $30|S|$. It remains to bound the number of isolated vertices.

There are at most $|S|$ isolated vertices u in $V(D(N,T))$ for which $\iota(u) \in S$. For the rest, If $u \in V(T)$ and $\phi(u) \in V(N)$ are both isolated vertices then we have $\iota(u) = \iota(\phi(u) \in \{\text{PAST}, \text{FUTURE}\}$ (as $\iota(u) \neq \iota(\phi(u))$ would imply u has out-degree 2 and so u and $\phi(u)$ are both redundant w.r.t σ. Similarly if v is an isolated network vertex with no $u \in V(T)$ for which $\phi(u) = v$, then v is redundant w.r.t σ. As

σ has no redundant vertices (since σ is well-behaved), it follows that every isolated vertex in $D(N,T)$ is either a tree vertex u with $\phi(u)$ not isolated, a network vertex $v = \phi(u)$ for which u is not isolated, or a vertex of $\iota^{-1}(S)$. As there are at most $30|S|$ non-isolated vertices, it follows that there are at most $30|S| + |S|$ isolated vertices.

Thus in total, $|V(D(N,T))| \leq 30|S| + 30|S| + |S| = 61|S| \in O(|S|)$.

An identical argument holds for a reconciliation μ. □

Lemma 30. *Let k be the width of the tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$. Then, the number of compact signatures for a bag (P,S,F) and the number of compact reconciliations for a Join bag can be upper-bounded by $2^{O(k^2)}$.*

Proof: Let $\sigma = (D(N,T), \phi, \iota)$ denote a compact signature for (P,S,F) (The arguments for a reconciliation μ are similar). By Lemma 29, $|V(D(N,T))| \in O(|S|)$ and, by the properties of a tree decomposition, $|S| \leq k+1$, implying $|V(D(N,T))| \in O(k)$. As such, an upper bound for the number of possible graphs $D(N,T)$ is $2^{O(k^2)} =: f_1(k)$.

For each vertex $u \in V(D(N,T))$, there are at most $(|S|-1)+2 \leq k+2$ possibilities for $\iota(u)$, as $\iota(u)$ is either a vertex in S other than u or one of the labels PAST and FUTURE (LEFT, RIGHT, and FUTURE for a reconciliations). Thus, the number of possible isolabelings for a given display graph $D(N,T)$ is $(k+3)^{O(k)} =: f_2(k)$

Now, to bound the number of possible embedding functions ϕ, observe that ϕ is fixed by (a) specifying $\phi(u)$ for every *tree vertex* u ($|V(N)|^{|V(T)|} \in k^{O(k)}$ possibilities) and (b) the set of arcs in N that appear in some path $\phi(uv)$ ($2^{|A(N)|} \in 2^{O(k^2)}$ possibilities) – indeed, if this set of arcs is chosen correctly, then it contains only one path from $\phi(u)$ to $\phi(v)$, which must be the path $\phi(uv)$. Thus, for any fixed display graph $D(N,T)$, the number of possible embedding functions is upper-bounded by $2^{O(k^2)} =: f_3(k)$.

Now, for any bag (P,S,F), the number of possible choices for $\sigma = (D(N,T), \phi, \iota)$ is bounded by $f(k) := f_1(k)f_2(k)f_3(k) = 2^{O(k^2)} \cdot 2^{O(k \log k)} \cdot 2^{O(k^2)} = 2^{O(k^2)}$. □

Lemma 31. *Algorithm 1 is correct and runs in $2^{O(k^2)} \cdot |A(N_{\text{IN}})|$ time, where $k = tw(N_{\text{IN}})$.*

Proof: First, note that, by Theorem 1, N_{IN} does not display T_{IN} unless $tw(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \leq 2k+1$, so we are safe to return FALSE if $tw(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})) \leq 2tw(N_{\text{IN}})+1$ in line 2 of Algorithm 1. Otherwise, for each bag x in a nice tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$, Algorithm 1 calculates the set CV_x of compact-valid signatures for x. In each case the set CV_x is calculated using the previously-calculated set CV_y for each child y of x. The correctness of this construction follows from Lemma 23 (for the Leaf bags), Lemma 24 (for Forget bags), Lemma 25 (for Introduce bags), and Corollary 2 (for Join bags). Finally, the algorithm returns TRUE if and only if there is a valid compact signature $(D(N,T), \phi, \iota)$ for the root bag of the tree decomposition, such that $\iota^{-1}(\text{FUTURE}) = \varnothing$. The correctness of this follows from Corollary 1.

To see the running time, first note that a nice, minimum-width tree decomposition of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ with $O(|V(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))|) = O(|A(N_{\text{IN}})|)$ bags can be found in $2^{O(tw(D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))^2)}|A(N_{\text{IN}})|$, that is, $2^{O(k^2)}|A(N_{\text{IN}})|$ time [Bod96, Klo94]. By Theorem 1, we may assume $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}})$ has treewidth at most $2k+1$ and, thus, $|S| \leq 2k+1$ for every bag (P,S,F) in the decomposition. Note that, computing any compact restriction of a signature σ can be done in polynomial time and, by Lemma 30, the number of such signatures $|CV_x|$ for a bag x is bounded by $2^{O(k^2)}$. It is, thus, evident that, for any bag $x = (P,S,F)$, the set CV_x can be computed in $2^{O(k^2)} \cdot k^{O(1)} = 2^{O(k^2)}$ time (see Algorithm 1). □

Lemma 31 immediately implies the following theorem.

Theorem 2. TREE CONTAINMENT *can be solved in* $2^{O(tw(N_{\text{IN}})^2)} \cdot |A(N_{\text{IN}})|$ *time.*

5 Future work

Before implementing our dynamic programming algorithm, one should first try to reduce the constant in the bound on the number of possible signatures as much as possible. Such reductions may be possible for instance by imposing further structural constraints on the signatures that need to be considered. It would also be important to find ways of generating valid signatures for one bag directly from the valid signatures if its child bag(s), rather than generating all possible signatures and then removing the invalid ones.

From a theoretical point of view, there are many opportunities for future work. First, there are multiple variants and generalizations of TREE CONTAINMENT that deserve attention, including non-binary inputs, and inputs consisting of two networks (i.e. where the task is to decide if a network is contained in a second network). For the latter problem our approach would have to be extensively modified, since our size-bound on the signatures heavily relies on T_{IN} being a tree. In the case of non-binary inputs, it is likely that a similar approach to the one in this paper would allow us to get a size-bound on the tree side of each signature. However, more work would be needed in order to prove a bound on the network side, and the number of possible embeddings. Note in particular that, under our current approach for deriving signatures from (partial) solutions, all neighbours of S are preserved. Without a bound on the degrees, the number of such vertices can be much larger than the treewidth of $D_{\text{IN}}(N_{\text{IN}}, T_{\text{IN}}))$. This can lead to an explosion in the length of 'compact' paths and the number of possible embeddings that one may need to consider for a single bag. Such explosions may be avoidable through clever bookkeeping, or it may be that they are unavoidable and can be used to force a $W[1]$-hardness reduction.

Second, a major open problem is whether the HYBRIDIZATION NUMBER problem is FPT with respect to the treewidth of the output network. Again there are different variants: rooted and unrooted, binary and non-binary, a fixed or unbounded number of input trees. For some applications, the definition of an embedding has to be relaxed (allowing, for example, multiple tree arcs embedded into the same network arc) [HMSW16, HLM21]. Other interesting candidate problems for treewidth-based algorithms include phylogenetic network drawing [KS20], orienting phylogenetic networks [HvIJ+19] and phylogenetic tree inference with duplications [vIJJ+19].

Finally, we believe that the approach taken in this paper (applying dynamic programming techniques on a tree decomposition of single graph representing all the input data, with careful attention given to the interaction between past and future) could potentially have applications outside of phylogenetics, in any context where the input to a problem consists of two or more distinct partially-labelled graphs that need to be reconciled.

Acknowledgements: Research of Leo van Iersel and Mark Jones was partially funded by Netherlands Organization for Scientific Research (NWO) Vidi grant 639.072.602 and KLEIN grant OCENW.KLEIN.125.

References

[ALS91] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12(2):308–340, 1991.

[BCKN15] Hans L. Bodlaender, Marek Cygan, Stefan Kratsch, and Jesper Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Inf. Comput.*, 243:86–111, 2015.

[BDD+16] Hans L Bodlaender, Pål Grønås Drange, Markus S Dregi, Fedor V Fomin, Daniel Lokshtanov, and Michał Pilipczuk. A $c^k n$ 5-approximation algorithm for treewidth. *SIAM Journal on Computing*, 45(2):317–378, 2016.

[BL06] David Bryant and Jens Lagergren. Compatibility of unrooted phylogenetic trees is FPT. *Theoretical Computer Science*, 351(3):296–302, 2006. Parameterized and Exact Computation.

[BMW18] Matthias Bentert, Josef Malík, and Mathias Weller. Tree containment with soft polytomies. In *SWAT'18*, volume 101, pages 9–1, 2018.

[Bod88] Hans L Bodlaender. Dynamic programming on graphs with bounded treewidth. In *ICALP'88*, pages 105–118. Springer, 1988.

[Bod96] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996.

[BPSC17] Julien Baste, Christophe Paul, Ignasi Sau, and Scornavacca Celine. Efficient FPT algorithms for (strict) compatibility of unrooted phylogenetic trees. *Bulletin of Mathematical Biology*, 79:920–938, 2017.

[BS07] Magnus Bordewich and Charles Semple. Computing the minimum number of hybridization events for a consistent evolutionary history. *Discrete Applied Mathematics*, 155(8):914–928, 2007.

[BSW20] Vincent Berry, Celine Scornavacca, and Mathias Weller. Scanning phylogenetic networks is NP-hard. In *SOFSEM'20*, pages 519–530. Springer, 2020.

[CFK+15] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*. Springer, 2015.

[CNP+11] Marek Cygan, Jesper Nederlof, Marcin Pilipczuk, Michal Pilipczuk, Joham MM van Rooij, and Jakub Onufry Wojtaszczyk. Solving connectivity problems parameterized by treewidth in single exponential time. In *FOCS'11*, pages 150–159. IEEE, 2011.

[Cou90] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990.

[Cou97] Bruno Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In *Handbook Of Graph Grammars And Computing By Graph Transformation: Volume 1: Foundations*, pages 313–400. World Scientific, 1997.

[EGHK21] Eduard Eiben, Robert Ganian, Thekla Hamm, and O-joung Kwon. Measuring what matters: A hybrid approach to dynamic programming with treewidth. *Journal of Computer and System Sciences*, 121:57–75, 2021.

[FHL08] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

[FHW80] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[FLP+20] Fedor V. Fomin, Daniel Lokshtanov, Fahad Panolan, Saket Saurabh, and Meirav Zehavi. *Hitting Topological Minors is FPT*, page 1317–1326. Association for Computing Machinery, New York, NY, USA, 2020.

[GGL+15] Philippe Gambette, Andreas D. M. Gunawan, Anthony Labarre, Stéphane Vialette, and Louxin Zhang. Solving the tree containment problem for genetically stable networks in quadratic time. In *IWOCA'15*, pages 197–208, 2015.

[GGL+18] Philippe Gambette, Andreas DM Gunawan, Anthony Labarre, Stéphane Vialette, and Louxin Zhang. Solving the tree containment problem in linear time for nearly stable phylogenetic networks. *Discrete Applied Mathematics*, 246:62–79, 2018.

[GHK+16] Robert Ganian, Petr Hliněný, Joachim Kneis, Daniel Meister, Jan Obdržálek, Peter Rossmanith, and Somnath Sikdar. Are there any good digraph width measures? *Journal of Combinatorial Theory, Series B*, 116:250–286, 2016.

[GKL15] A Grigoriev, S Kelk, and L Lekic. On low treewidth graphs and supertrees. *Journal of Graph Algorithms and Applications*, 19(1):325–343, 2015.

[GKMW11] Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *STOC'11*, pages 479–488, 2011.

[GLZ16] Andreas DM Gunawan, Bingxin Lu, and Louxin Zhang. A program for verification of phylogenetic network models. *Bioinformatics*, 32(17):i503–i510, 2016.

[Gun18] Andreas DM Gunawan. Solving the tree containment problem for reticulation-visible networks in linear time. In *AlCoB'18*, pages 24–36. Springer, 2018.

[GYZ19] Andreas DM Gunawan, Hongwei Yan, and Louxin Zhang. Compression of phylogenetic networks and algorithm for the tree containment problem. *Journal of Computational Biology*, 26(3):285–294, 2019.

[HLM21] Katharina T Huber, Simone Linz, and Vincent Moulton. The rigid hybrid number for two phylogenetic trees. *Journal of Mathematical Biology*, 82(5):1–29, 2021.

[HMSW16] Katharina T Huber, Vincent Moulton, Mike Steel, and Taoyang Wu. Folding and unfolding phylogenetic trees and networks. *Journal of Mathematical Biology*, 73(6):1761–1780, 2016.

[HvIJ+19] Katharina T Huber, Leo van Iersel, Remie Janssen, Mark Jones, Vincent Moulton, Yukihiro Murakami, and Charles Semple. Rooting for phylogenetic networks. *arXiv preprint arXiv:1906.07430*, 2019.

[IKS+18] Leo van Iersel, Steven Kelk, Georgios Stamoulis, Leen Stougie, and Olivier Boes. On unrooted and root-uncertain variants of several well-known phylogenetic network problems. *Algorithmica*, 80(11):2993–3022, 2018.

[ISS10] Leo van Iersel, Charles Semple, and Mike Steel. Locating a tree in a phylogenetic network. *Information Processing Letters*, 110(23):1037–1043, 2010.

[JJK+19] R. Janssen, M. Jones, S. Kelk, G. Stamoulis, and T. Wu. Treewidth of display graphs: bounds, brambles and applications. *Journal oof Graph Algorithms and Applications*, 23:715–743, 2019.

[JM21] Remie Janssen and Yukihiro Murakami. On cherry-picking and network containment. *Theoretical Computer Science*, 856:121–150, 2021.

[Klo94] Ton Kloks. *Treewidth: computations and approximations*. Springer, 1994.

[KNTX08] Iyad A Kanj, Luay Nakhleh, Cuong Than, and Ge Xia. Seeing the trees and their branches in the network is hard. *Theoretical Computer Science*, 401(1-3):153–164, 2008.

[Kor21] Tuukka Korhonen. A single-exponential time 2-approximation algorithm for treewidth. In *FOCS'21*, pages 184–192, 2021.

[KS20] Jonathan Klawitter and Peter Stumpf. Drawing tree-based phylogenetic networks with minimum number of crossings. In *International Symposium on Graph Drawing and Network Visualization*, pages 173–180. Springer, 2020.

[KSW18] Steven Kelk, Georgios Stamoulis, and Taoyang Wu. Treewidth distance on phylogenetic trees. *Theoretical Computer Science*, 731:99–117, 2018.

[KvSW16] Steven Kelk, Leo van Iersel, Celine Scornavacca, and Mathias Weller. Phylogenetic incongruence through the lens of monadic second order logic. *Journal of Graph Algorithms and Applications*, 20(2):189–215, 2016.

[SW21] Céline Scornavacca and Mathias Weller. Treewidth-based algorithms for the small parsimony problem on networks. In *WABI'21*, volume 201 of *LIPIcs*, pages 6:1–6:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[vIJJ+19] Leo van Iersel, Remie Janssen, Mark Jones, Yukihiro Murakami, and Norbert Zeh. Polynomial-time algorithms for phylogenetic inference problems involving duplication and reticulation. *IEEE/ACM transactions on computational biology and bioinformatics*, 17(1):14–26, 2019.

[vIKL+16] Leo van Iersel, Steven Kelk, Nela Lekic, Chris Whidden, and Norbert Zeh. Hybridization number on three rooted binary trees is ept. *SIAM Journal on Discrete Mathematics*, 30(3):1607–1631, 2016.

[vIKS16] Leo van Iersel, Steven Kelk, and Celine Scornavacca. Kernelizations for the hybridization number problem on multiple nonbinary trees. *Journal of Computer and System Sciences*, 82(6):1075–1089, 2016.

[vIL13] Leo van Iersel and Simone Linz. A quadratic kernel for computing the hybridization number of multiple trees. *Information Processing Letters*, 113(9):318–323, 2013.

[Wel18] Mathias Weller. Linear-time tree containment in phylogenetic networks. In *RECOMB-CG'18*, volume 11183 of *LNCS*, pages 309–323. Springer, 2018.