A Complete Characterisation of the Classification Tree Problem

Pierre Flener[1] and Xavier Lorca[2]

[1] Department of Information Technology, and
The Linnaeus Centre for Bioinformatics
Uppsala University, Box 337, SE – 751 05 Uppsala, Sweden
Pierre.Flener@it.uu.se

[2] École des Mines de Nantes, LINA UMR CNRS 6241
FR – 44307 Nantes Cedex 3, France
Xavier.Lorca@emn.fr

Abstract. Finding a classification tree over a given set of elements that is compatible with a given family of classification trees over subsets of that set is a common problem in many application areas, such as the historical analysis of languages, the theory of relational databases, and phylogenetic supertree construction. We present a constraint programming approach to this problem. First, we introduce a natural and compact graph representation of a family of classification trees. Second, we provide a complete filtering algorithm for the classification tree problem, based on this normal form.

1 Introduction

In real life, the ordering of elements is a common problem. A given set of elements can be partially ordered in various ways according to criteria. A standard classification of elements is represented by a tree structure, such as XML documents. In this way, a recurrent question is:

> Which classification tree over a given set of elements unifies a given family of classification trees over subsets of that set?

This is the case for the analysis of historical materials (like texts, artifacts, etc) leading to various classifications (of languages [5], etc), the theory of relational databases [1], and the analysis of evolutionary relationships between species [3]. In particular, the latter application was previously studied by the constraint programming community [7, 8, 2]. We show in the following the first complete characterisation of the phylogenetic problem and a filtering algorithm detecting all the infeasible relationships between species.

Formally, this problem can be interpreted in graph theory. Indeed, each classification tree can be represented by a directed tree (more precisely, an *anti-arborescence*, whose arcs are oriented from the leaves toward the root) where only the leaves (denoting the classified elements) are labelled and the internal nodes are anonymous. The problem then lies in finding a directed tree $\mathcal{T}$ such that each given classification tree is homeomorphic to some subtree of $\mathcal{T}$ (that is, if they can be transformed into each other by adding or removing nodes of in-degree 1).

Our running example is given in Figure 1, where Figure 1a depicts a family $(\mathcal{T}_{\mathcal{U}})$ of three classification trees (the trees are binary, but this is not necessary). Figures 1b and 1c represent the construction steps of the so-called *normal form* of the tree family; the definition, properties, and construction of the normal form are detailed in Section 3: for convenience all the sub-figures are together here, but the reader can ignore these two sub-figures for now. Figure 1d represents a tree, denoted $\mathcal{T}$, that respect all the information provided by the normal form. Consider the set of sources (or leaves) in $\mathcal{T}_1$, denoted by $\mathcal{S}_1 = \{A, D, E\}$ and consider the subgraph (depicted by the bold arcs of Figure 1d) $\mathcal{T}(\mathcal{S}_1)$ of $\mathcal{T}$ induced by $\mathcal{S}_1$ and its descendants in $\mathcal{T}$: we have that $\mathcal{T}(\mathcal{S}_1)$ and $\mathcal{T}_1$ are homeomorphic. With the same reasoning, $\mathcal{T}_2$ and $\mathcal{T}_3$ are also homeomorphic with subtrees of $\mathcal{T}$. Finally, Figure 1e depicts a solution, which is a complete tree (defined in Section 2.1), to the corresponding classification tree problem. This complete tree is obtained from $\mathcal{T}$ of Figure 1e by removing nodes of in-degree 1. In the following, this complete tree is said *compatible* with the family (see Definition 1).

After defining, in Section 2, the underlying concepts of the classification tree problem, the organisation and contributions of this paper are as follows:

- We introduce a natural and compact graph representation of a family of classification trees. Moreover, we show that if such a normal form has a circuit, then the related classification tree problem has no solutions. While the computation of this intermediate representation can technically be omitted, we advocate it as an alternative output to the classification tree problem; witness the recent interest in phylogenetic super-networks rather than supertrees [9]. (Section 3)
- We propose a novel constraint-programming (CP) approach to enumerating all the trees that are compatible with a given family of trees. The filtering algorithm for the underlying new global constraint is complete, hence this is probably the first CP approach to the problem that provably takes polynomial time. (Section 4)
- We illustrate the application of our approach to the phylogenetic supertree problem, and theoretically compare it to the state-of-the-art approaches. (Section 5)

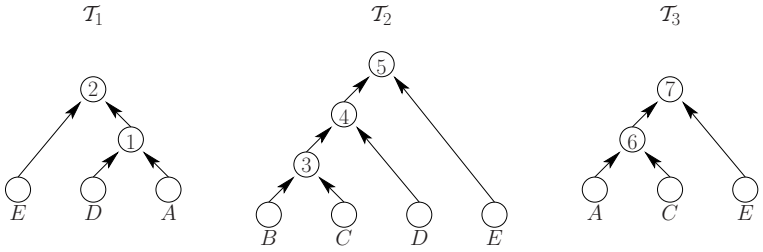
Finally, Section 6 concludes and provides some directions for future work.

2 Preliminaries

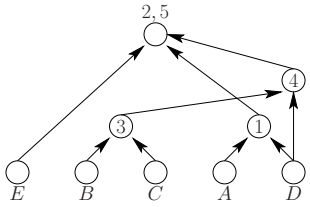
We now introduce the underlying concepts of the classification tree problem. Section 2.1 shows how to represent a classification of elements with a directed tree, while Section 2.2 provides a formal definition of the compatibility of a classification tree with a family of classification trees.

2.1 Graph Representation of a Classification Tree

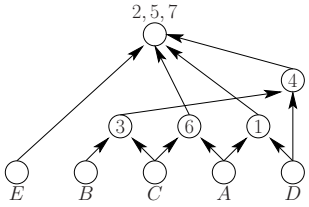
A *classification tree* $\mathcal{T}$ is defined over a set $\mathcal{S}$ of classified elements, which are the leaves of $\mathcal{T}$. For any non-empty subset $\mathcal{E}$ of $\mathcal{S}$, there exists a single node in $\mathcal{T}$ that is (on the paths to the root) the *first common descendant* (fcd) of the elements of $\mathcal{E}$, denoted by $\Phi(\mathcal{E})$. Note that if $\mathcal{E}$ contains a single element e of $\mathcal{S}$, then $\Phi(\mathcal{E}) = e$. If $\mathcal{E} = \mathcal{S}$, then $\Phi(\mathcal{E})$ is the single root of the tree $\mathcal{T}$. Given two subsets $\mathcal{E}_1 \subseteq \mathcal{E}_2 \subseteq \mathcal{S}$, we write



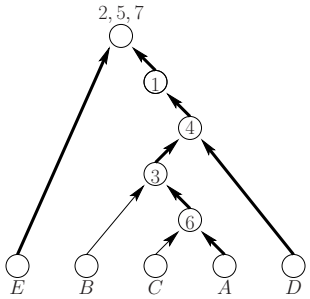
(a) A family $(\mathcal{T}_\mathcal{U})$ of three trees



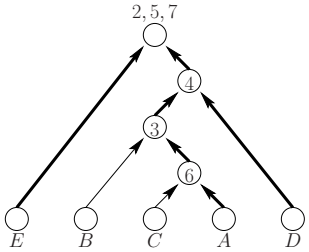
(b) $\mathcal{G}_\mathcal{U} = \mathcal{T}_1 + \mathcal{T}_2$



(c) $\mathcal{G}_\mathcal{U} = \mathcal{T}_1 + \mathcal{T}_2 + \mathcal{T}_3$



(d) Tree $\mathcal{T}$ compatible with $(\mathcal{T}_\mathcal{U})$



(e) A compatible complete tree

Fig. 1: A sample classification tree problem.

$\Phi(\mathcal{E}_1) \preceq \Phi(\mathcal{E}_2)$ and say that the fcd of $\mathcal{E}_1$ *precedes* the fcd of $\mathcal{E}_2$. Formally, let $\mathcal{P}(\mathcal{S})$ be the set of subsets of $\mathcal{S}$ and let $\mathcal{N}$ be the set of nodes in $\mathcal{T}$ (including the elements of $\mathcal{S}$). The surjective function $\Phi : \mathcal{P}(\mathcal{S}) \to \mathcal{N}$ returns, for a given non-empty subset $\mathcal{E}$ of $\mathcal{S}$, the single node n of $\mathcal{N}$ that is the fcd of the elements of $\mathcal{E}$. Similarly, given two subsets $\mathcal{E}_1$ and $\mathcal{E}_2$ of $\mathcal{S}$ such that $\Phi(\mathcal{E}_1)$ and $\Phi(\mathcal{E}_2)$ denote the same node of $\mathcal{T}$, we write $\Phi(\mathcal{E}_1) = \Phi(\mathcal{E}_2)$ and say that $\Phi(\mathcal{E}_1)$ and $\Phi(\mathcal{E}_2)$ are *identical*. Note that this does not imply that $\mathcal{E}_1$ and $\mathcal{E}_2$ are the same subsets. In a classification tree $\mathcal{T}$, each node is the fcd of several subsets of $\mathcal{S}$. The function $\Phi^{-1} : \mathcal{N} \to \mathcal{P}(\mathcal{S})$ returns, for a given node n, the maximal subset (under inclusion) of $\mathcal{S}$ for which n is the fcd. Finally, a classification tree $\mathcal{T}$ is completely defined by the partially ordered set (poset) $(\mathcal{N}, \preceq)$. Indeed, for any three nodes x, y, z of $\mathcal{N}$:

– $\preceq$ is reflexive: $x \preceq x$. Any node of $\mathcal{T}$ is its own fcd.

– $\preceq$ is anti-symmetric: $x \preceq y \wedge y \preceq x \Rightarrow x = y$. Let $\mathcal{E} = \Phi^{-1}(x)$ and $\mathcal{F} = \Phi^{-1}(y)$. Hence $\mathcal{E} \subseteq \mathcal{F}$ and $\mathcal{F} \subseteq \mathcal{E}$, thus $\mathcal{E} = \mathcal{F}$, and by definition $x = \Phi(\mathcal{E}) = \Phi(\mathcal{F}) = y$.
– $\preceq$ is transitive: $x \preceq y \wedge y \preceq z \Rightarrow x \preceq z$. Let $\mathcal{E} = \Phi^{-1}(x)$, $\mathcal{F} = \Phi^{-1}(y)$, and $\mathcal{G} = \Phi^{-1}(z)$. Hence $\mathcal{E} \subseteq \mathcal{F}$ and $\mathcal{F} \subseteq \mathcal{G}$, thus $\mathcal{E} \subseteq \mathcal{G}$, and by definition $x \preceq z$.

A classification tree $\mathcal{T}$ is naturally represented by a directed graph (digraph) defined by the transitive reduction of the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ with arc set $\mathcal{A} = \{(x, y) \mid x \in \mathcal{N} \wedge y \in \mathcal{N} \wedge x \neq y \wedge x \preceq y\}$, where $\preceq$ is defined as above for the function Φ that associates to each non-empty set of classified elements its first common descendant.

Property 1 (Directed Complete Tree) *The digraph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ of a classification tree $\mathcal{T}$ is a directed complete tree (whose arcs are oriented from the leaves to the root).*

Proof. Recall that a digraph is a tree if and only if it is a connected graph, with a single sink node, without any circuits, and without any transitive arcs. In our context:

– There exists a single sink node in $\mathcal{G}$ because $\Phi(\mathcal{S})$ is unique. As a consequence, $\mathcal{G}$ is connected.
– The pair $(\mathcal{N}, \preceq)$ is a poset, as shown above. In particular, there do not exist any three distinct nodes $x, y, z \in \mathcal{N}$ such that $x \preceq y$, $y \preceq z$, and $z \preceq x$, hence $\mathcal{G}$ does not contain any circuits.
– $\mathcal{G}$ is defined by the transitive reduction of some graph, hence it does not contain any transitive arcs.
– Every internal node of $\mathcal{G}$ has an in-degree of at least 2, hence $\mathcal{G}$ is complete. □

Note that the classified elements $\mathcal{S}$ of a classification tree $\mathcal{T}$ correspond to the source (or leaf) nodes of its digraph representation $\mathcal{G}$. Similarly, the internal nodes of $\mathcal{T}$ correspond to the internal nodes of $\mathcal{G}$. Finally, for a non-empty subset $\mathcal{E}$ of the classified elements $\mathcal{S}$ of $\mathcal{T}$, the node $\Phi(\mathcal{E})$ corresponds to the first common descendant (in the direction of the arcs) of the source nodes corresponding to $\mathcal{E}$ in $\mathcal{G}$. In the following, we gloss over the difference between classification trees and their representation as directed complete trees, and we often simply speak about trees rather than directed complete trees or classification trees.

2.2 The Classification Tree Problem

A *subtree* $\mathcal{T}_s = (\mathcal{U}_s, \mathcal{V}_s)$ of a tree $\mathcal{T} = (\mathcal{U}, \mathcal{V})$ is a subgraph of $\mathcal{T}$, i.e., $\mathcal{U}_s \subseteq \mathcal{U}$ and $\mathcal{V}_s \subseteq \mathcal{V}$. Also, two digraphs $\mathcal{G}$ and $\mathcal{H}$ are *homeomorphic* (or *topologically equivalent*), denoted by $\mathcal{G} \approx \mathcal{H}$, if they can be transformed into each other by adding or removing nodes of in-degree 1. Finally, two digraphs are *isomorphic* if there is a one-to-one correspondence between their nodes, and if there is an arc between two nodes of one graph if and only if there is an arc between the two corresponding nodes in the other graph. Two digraphs are thus homeomorphic if and only if they become isomorphic by adding or removing nodes of in-degree 1, which does not change the topology of the digraph, as the set of paths between each pair of nodes is preserved.

In the context of our graph interpretation of the classification tree problem, a tree preserving all the trees of a family $(\mathcal{T}_\mathcal{U})$ of trees is defined by (also see Figure 1):

Definition 1 (Compatible Tree). *A tree $\mathcal{T}$ is* compatible *with a tree family $(\mathcal{T}_\mathcal{U})$ if and only if there exists a homeomorphic subtree of $\mathcal{T}$ for each tree of $(\mathcal{T}_\mathcal{U})$.*

Throughout the paper, for a tree $\mathcal{T}_i$ of a tree family $(\mathcal{T}_\mathcal{U})$, with $i \in \mathcal{U}$, the node set $\mathcal{N}_i$, the source (or leaf) set $\mathcal{S}_i$, the arc set $\mathcal{A}_i$, and the first-common-descendant function Φ_i of $\mathcal{T}_i$ are often implicit.

3 Normal Form of a Family of Classification Trees

We propose a novel compact representation, called the *normal form*, of the information of a given family $(\mathcal{T}_\mathcal{U})$ of classification trees. Toward this, Section 3.1 presents an equivalence relation between the nodes of distinct trees of $(\mathcal{T}_\mathcal{U})$. Section 3.2 defines the normal form, establishes its properties, and constructively shows how to compute it. Section 3.3 shows a necessary and sufficient condition for the classification tree problem, based on the normal form of the tree family.

3.1 Equivalent Nodes

Intuitively, for any shared subset $\mathcal{E}$ of classified elements of two trees $\mathcal{T}_i$ and $\mathcal{T}_j$, the first common descendants of $\mathcal{E}$ in $\mathcal{T}_i$ and $\mathcal{T}_j$ represent in fact the *same* node $\Phi(\mathcal{E})$ in a tree compatible with $\mathcal{T}_i$ and $\mathcal{T}_j$. We recall that only the leaves are labelled in a classification tree, hence the leaves of distinct trees with identical labels will be considered equivalent.

Consider a set $\mathcal{E}$ of sources shared by a subset $\mathcal{R}$ of trees of a family $(\mathcal{T}_\mathcal{U})$. The first-common-descendant relation over $\mathcal{R}$ enforces that the first common descendant of $\mathcal{E}$ in any tree compatible with $(\mathcal{T}_\mathcal{U})$ is unique, if it exists. Then, the first common descendants of $\mathcal{E}$ in each tree of $\mathcal{R}$ are all pairwise equivalent. Moreover, the reasoning can be improved. Indeed, consider a tree $\mathcal{T}_j$ of $(\mathcal{T}_\mathcal{U})$ and an internal node x_j of $\mathcal{T}_j$. The maximum (under node inclusion) subset of sources for which x_j is the first common descendant is $\Phi_j^{-1}(x_j)$. Consider two subsets $\mathcal{E}_j^1, \mathcal{E}_j^2 \subseteq \Phi_j^{-1}(x_j)$. If there exist two trees $\mathcal{T}_i$ and $\mathcal{T}_k$ of $(\mathcal{T}_\mathcal{U})$ and two nodes x_i and x_k respectively therein such that $\Phi_i^{-1}(x_i) = \mathcal{E}_j^1$ and $\Phi_k^{-1}(x_k) = \mathcal{E}_j^2$, then node x_j has to be merged with x_i and x_k in any tree compatible with $(\mathcal{T}_\mathcal{U})$. Thus, $\{x_i, x_j, x_k\}$ are pairwise equivalent. These two observations lead to the following definition:

Definition 2 (Equivalent Nodes). *Given a tree family $(\mathcal{T}_\mathcal{U})$, two nodes x_i and x_j of respectively trees $\mathcal{T}_i$ and $\mathcal{T}_j$ of $(\mathcal{T}_\mathcal{U})$ are equivalent (denoted by $x_i \equiv x_j$) if and only if one of the following conditions hold:*

1. *There exists a non-empty shared subset $\mathcal{E}$ of sources in $\mathcal{T}_i$ and $\mathcal{T}_j$ such that $\Phi_i(\mathcal{E}) = x_i$ and $\Phi_j(\mathcal{E}) = x_j$,*
2. *There exist a tree $\mathcal{T}_k$ of $(\mathcal{T}_\mathcal{U})$ and a node x_k of $\mathcal{T}_k$ such that $x_i \equiv x_k$ and $x_k \equiv x_j$.*

For example, in Figure 1a, the set of elements $\mathcal{E} = \{D, E\}$ is shared by $\mathcal{T}_1$ and $\mathcal{T}_2$ then, Definition 2 ensures $(\Phi_1(\mathcal{E}) = 2) \equiv (\Phi_2(\mathcal{E}) = 5)$. However, this is a straightforward example. Indeed, consider the set $\mathcal{E}_1 = \{A, E\}$ of $\mathcal{T}_1$ and the set $\mathcal{E}_2 = \{B, C, E\}$

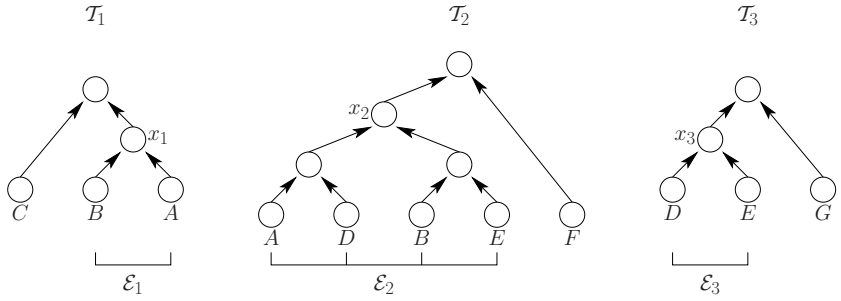


Fig. 2: Transitivity of the equivalence ($\equiv$) relation.

of $\mathcal{T}_2$. We respectively have $\Phi_1(\mathcal{E}) = \Phi_1(\mathcal{E}_1) = 2$ in $\mathcal{T}_1$ and $\Phi_2(\mathcal{E}) = \Phi_2(\mathcal{E}_2) = 5$ in $\mathcal{T}_2$. Then, we also have $\Phi_1(\mathcal{E}_1) \equiv \Phi_2(\mathcal{E}_2)$.

Let $\mathcal{N}$ be the set of all nodes in a family $(\mathcal{T}_\mathcal{U})$ of trees. Then $\equiv$ is an equivalence relation over $\mathcal{N}$. Indeed, for any three nodes x, y, z of $\mathcal{N}$:

- $\equiv$ is reflexive: $x \equiv x$. Take $\mathcal{T}_i = \mathcal{T}_j$ in Definition 2 and $\mathcal{E}$ such that $\Phi_i(\mathcal{E}) = x$.
- $\equiv$ is symmetric: $x \equiv y \Rightarrow y \equiv x$. This follows from the interchangeability of $\mathcal{T}_i$ and $\mathcal{T}_j$ in Definition 2.
- $\equiv$ is transitive: $x \equiv y \wedge y \equiv z \Rightarrow x \equiv z$. This is due to item 2 of Definition 2.

The example, depicted by Figure 2, illustrates the transitivity of the equivalence relation. In $\mathcal{T}_1$ we have $\Phi_1(\mathcal{E}_1) = x_1$, while in $\mathcal{T}_2$ we have $\Phi_2(\mathcal{E}_1) = \Phi_2(\mathcal{E}_2) = x_2$. Similarly, in $\mathcal{T}_3$ we have $\Phi_3(\mathcal{E}_3) = x_3$, while in $\mathcal{T}_2$ we have $\Phi_2(\mathcal{E}_3) = \Phi_2(\mathcal{E}_2) = x_2$. Then we have $x_1 \equiv x_2$ and $x_2 \equiv x_3$, so x_2 has to be merged with x_1 and x_3 in any compatible tree. This means that $x_1 \equiv x_3$.

Let $\mathcal{N}^{\equiv}$ be the set of nodes produced by contracting all the equivalent nodes of $\mathcal{N}$. The next sub-section essentially shows that a tree $\mathcal{T}$ that is compatible with a tree family $(\mathcal{T}_\mathcal{U})$ is a tree that respects the poset $(\mathcal{N}^{\equiv}, \preceq)$.

3.2 Modelling the Information of a Family of Classification Trees

We now propose a novel compact way to represent the information of a family of classification trees. We aim at a representation that is unique, as well as minimal in terms of internal nodes and arcs, without losing any information in the given trees.

Definition 3 (Normal Form of a Tree Family). *The* normal form *of a tree family $(\mathcal{T}_\mathcal{U})$ is the minimal (under node inclusion) digraph $\mathcal{G}_\mathcal{U}$ without transitive arcs such that there exists a homeomorphic subgraph of $\mathcal{G}_\mathcal{U}$ for each tree of $(\mathcal{T}_\mathcal{U})$.*

For example, Figure 1b depicts the normal form of the family $\{\mathcal{T}_1, \mathcal{T}_2\}$. Next, Figure 1c depicts the final normal form of the family represented by Figure 1a. The equivalent nodes 2, 5, and 7, of trees $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$, respectively, are merged into a single node of the normal form $\mathcal{G}_\mathcal{U}$.

Computing the normal form of a tree family $(\mathcal{T}_\mathcal{U})$ can be done by iteratively merging the trees of $(\mathcal{T}_\mathcal{U})$ into an initially empty directed graph $\mathcal{G}_\mathcal{U}$. During iteration k, the tree $\mathcal{T}_k$ of $(\mathcal{T}_\mathcal{U})$ is merged into $\mathcal{G}_\mathcal{U}$. First, each node of $\mathcal{T}_k$ is added to $\mathcal{G}_\mathcal{U}$ except if it is equivalent to an existing node of $\mathcal{G}_\mathcal{U}$ (according to Definition 2). Second, each arc of $\mathcal{T}_k$ is added to $\mathcal{G}_\mathcal{U}$ except if it is transitive, and induced transitive arcs are removed. At the end, a single sink node succeeding all the sink nodes of $\mathcal{G}_\mathcal{U}$ is added to $\mathcal{G}_\mathcal{U}$. We now show that this merging procedure is correct:

Lemma 1 (Correctness of Normal-Form Computation). *Let $\mathcal{T}_k$ be a tree of $(\mathcal{T}_\mathcal{U})$ not merged yet into the normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$. For any node $u \in \mathcal{N}_k$, if there exists a node $v \in \mathcal{N}_\mathcal{U}$ such that $u \equiv v$, then v is unique in $\mathcal{N}_\mathcal{U}$.*

Proof. Assume there exist two nodes $v, v' \in \mathcal{N}_\mathcal{U}$, with $v \neq v'$, such that $u \equiv v$ and $u \equiv v'$. By the transitivity of $\equiv$ (proven above), we then have $v \equiv v'$. Since v and v' necessarily stem from two distinct trees of $(\mathcal{T}_\mathcal{U})$ (as no two nodes of a tree can be equivalent), one of these two nodes is merged with the other one during the merging of the second of those trees into $\mathcal{G}_\mathcal{U}$ (as the given trees are merged sequentially), hence we have a contradiction with the assumed $v \neq v'$. □

Now we are in position to introduce the two properties that make $\mathcal{G}_\mathcal{U}$ a normal form of the family $(\mathcal{T}_\mathcal{U})$, namely the uniqueness in $\mathcal{G}_\mathcal{U}$ of the first common descendant of any subset of the classified elements, and the uniqueness and minimality of $\mathcal{G}_\mathcal{U}$ itself. Toward this, we first naturally extend the tree functions $\Phi_i : \mathcal{P}(\mathcal{S}_i) \to \mathcal{N}_i$ and $\Phi_i^{-1} : \mathcal{N}_i \to \mathcal{P}(\mathcal{S}_i)$ on the given trees $\mathcal{T}_i$ of $(\mathcal{T}_\mathcal{U})$ to graph functions $\Phi_\mathcal{U} : \mathcal{P}(\mathcal{S}_\mathcal{U}) \to \mathcal{P}(\mathcal{N}_\mathcal{U})$ and $\Phi_\mathcal{U}^{-1} : \mathcal{P}(\mathcal{N}_\mathcal{U}) \to \mathcal{P}(\mathcal{S}_\mathcal{U})$. However, notice that for a set $\mathcal{E} \in \mathcal{P}(\mathcal{S}_\mathcal{U})$, if $\mathcal{E} \not\subseteq \mathcal{S}_k$ for every $\mathcal{T}_k$ in $(\mathcal{T}_\mathcal{U})$, then $\Phi_\mathcal{U}(\mathcal{E})$ is the empty set.

We also introduce a new notation defining the merged nodes in the normal form. For a node i of a tree $\mathcal{T}_k$ of tree family $(\mathcal{T}_\mathcal{U})$, let $\Omega(i)$ denote the node u of the normal form $\mathcal{G}_\mathcal{U}$ to which i is equivalent, so that $\Omega^{-1}(u)$ denotes the set of nodes in the trees of $(\mathcal{T}_\mathcal{U})$ that are equivalent to u. Moreover, let $\mathcal{C}_k$ denote the transitive closure of tree $\mathcal{T}_k$, and let $\mathcal{C}_\mathcal{U}$ denote the transitive closure of the normal form. It will be important that these transitive closures are not reflexive, so that $u \notin \mathcal{C}_k(u)$ for any $\mathcal{T}_k$ and $u \notin \mathcal{C}_\mathcal{U}(u)$.

Property 2 (Uniqueness of First Common Descendants in Normal Form) *For each node u of $\mathcal{N}_\mathcal{U}$, the nodes of $\Omega^{-1}(u)$ form an equivalence class for the relation $\equiv$.*

Proof. A node w of a tree $\mathcal{T}_k$ in the family $(\mathcal{T}_\mathcal{U})$ cannot belong to two distinct sets $\Omega^{-1}(u)$ and $\Omega^{-1}(v)$ with $u, v \in \mathcal{N}_\mathcal{U}$ and $u \neq v$. Indeed, $w \in \Omega^{-1}(u)$ implies $u \equiv w$ and $w \in \Omega^{-1}(v)$ implies $v \equiv w$, hence, by the transitivity of $\equiv$, we have $u \equiv w$, which leads to $\Omega^{-1}(u) = \Omega^{-1}(v)$: contradiction. □

Property 3 (Uniqueness and Minimality of Normal Form) *The normal form $\mathcal{G}_\mathcal{U}$ of a tree family $(\mathcal{T}_\mathcal{U})$ is unique and minimal (under node inclusion).*

Proof. In the normal form $\mathcal{G}_\mathcal{U}$ of a tree family $(\mathcal{T}_\mathcal{U})$, any pair of equivalent nodes (according to Definition 2) are merged into a single node of $\mathcal{G}_\mathcal{U}$ (because they belong to an equivalence class for the relation $\equiv$). Hence two nodes of $\mathcal{G}_\mathcal{U}$ cannot be merged by Definition 2 (as a direct consequence of Lemma 1), and consequently $\mathcal{G}_\mathcal{U}$ is unique and minimal under node inclusion. □

In a directed complete tree, the number of internal nodes is at most the number of sources minus one. We denote by ℓ the number of distinct sources in the tree family $(\mathcal{T}_\mathcal{U})$. The number of nodes in the normal form $\mathcal{G}_\mathcal{U}$ is $O(\ell)$. Assume that the information Φ_k of a tree $\mathcal{T}_k \in (\mathcal{T}_\mathcal{U})$ is known and that $\Phi_\mathcal{U}$ is dynamically maintained for $\mathcal{G}_\mathcal{U}$. For each node u of each tree $\mathcal{T}_k$ in $(\mathcal{T}_\mathcal{U})$, we have to check the equivalence of u to every node v of the current $\mathcal{G}_\mathcal{U}$, using Definition 2 according to $\Phi_k^{-1}(u)$ and $\Phi_\mathcal{U}^{-1}(v)$. This can be done in $O(\ell^2)$ time. Moreover, the number of arcs of a tree is the number of nodes minus 1. Hence the addition of the arcs of a tree $\mathcal{T}_k$ of $(\mathcal{T}_\mathcal{U})$ takes $O(\ell^2)$ time. Indeed, checking the transitivity of each arc (u, v) of a tree $\mathcal{T}_k$ in the normal form and removing induced transitive arcs can be done by checking whether u and v are equivalent to existing nodes of $\mathcal{G}_\mathcal{U}$. If so, then two cases are distinguished. First, if $\Omega(v) \in \mathcal{C}_\mathcal{U}(\Omega(u))$, then the arc (u, v) is transitive and is not added to $\mathcal{G}_\mathcal{U}$. Second, if $\Omega(v) \notin \mathcal{C}_\mathcal{U}(\Omega(u))$ and there exists a node w in $\mathcal{G}_\mathcal{U}$ such that $w \in \mathcal{C}_\mathcal{U}(\Omega(v))$ (respectively $\Omega(u) \in \mathcal{C}_\mathcal{U}(w)$), then $(\Omega(u), w)$ (respectively $(w, \Omega(v))$) is a transitive arc and has to be removed from $\mathcal{G}_\mathcal{U}$. In conclusion, the computation of the normal form takes $O(\ell^2)$ time.

Each tree $\mathcal{T}_k = (\mathcal{N}_k, \mathcal{A}_k)$ of $(\mathcal{T}_\mathcal{U})$ induces some infeasible relationships between its nodes. Any two nodes $u, v \in \mathcal{N}_k$ such that the arc (u, v) is not in the transitive closure of $\mathcal{A}_k$ does not respect the partial order induced by $\mathcal{T}_k$. We say that the arc (u, v) is *impossible* for $\mathcal{T}_k$. Formally:

$$\forall u \in \mathcal{N}_k : \mathcal{R}_k(u) = \{v \in \mathcal{N}_k \mid v \notin \mathcal{C}_k(u)\}$$

This impossibility relation is naturally extended to the normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$. An arc (u, v) is *impossible* for $\mathcal{G}_\mathcal{U}$ if there exists a tree $\mathcal{T}_k$ in $(\mathcal{T}_\mathcal{U})$ such that there exist nodes $i \in \Omega^{-1}(u)$ and $j \in \Omega^{-1}(v)$, with $i, j \in \mathcal{N}_k$, such that $j \in \mathcal{R}_k(i)$. Moreover, the sources $\mathcal{S}_\mathcal{U} = \bigcup_{k \in \mathcal{U}} \mathcal{S}_k$ are pairwise impossible for $\mathcal{G}_\mathcal{U}$. Formally:

$$\forall u \in \mathcal{N}_\mathcal{U} : \mathcal{R}_\mathcal{U}(u) = \mathcal{S}_\mathcal{U} \cup \{v \in \mathcal{N}_\mathcal{U} \mid \exists k \in \mathcal{U} : \exists i, j \in \mathcal{N}_k : i \in \Omega^{-1}(u) \wedge j \in \Omega^{-1}(v) \wedge j \in \mathcal{R}_k(i)\}$$

Property 4 *If $w \in \mathcal{R}_\mathcal{U}(u)$ and $v \in \mathcal{C}_\mathcal{U}(u)$, then $w \in \mathcal{R}_\mathcal{U}(v)$.*

Proof. Let $u \in \mathcal{N}_\mathcal{U}$ and $v \in \mathcal{C}_\mathcal{U}(u)$. If there is a node $w \in \mathcal{R}_\mathcal{U}(u)$, then we have $w \notin \mathcal{C}_\mathcal{U}(u)$. According to $v \in \mathcal{C}_\mathcal{U}(u)$, the nodes u, v, w cannot be ordered on a single path in any compatible tree, so $w \notin \mathcal{C}_\mathcal{U}(v) \cup \mathcal{C}_\mathcal{U}(u)$ because $\mathcal{C}_\mathcal{U}(v) \subseteq \mathcal{C}_\mathcal{U}(u)$. Hence $w \notin \mathcal{C}_\mathcal{U}(v)$ and as a consequence $w \in \mathcal{R}_\mathcal{U}(v)$. □

3.3 Necessary and Sufficient Condition for the Classification Tree Problem

We now provide a complete characterisation of the classification tree problem based on the existence of an acyclic normal form for the given family $(\mathcal{T}_\mathcal{U})$ of classification trees. Toward this, we first need an important lemma, ensuring that for each non-sink node of the normal form, there exists at least one successor node that contradicts neither the impossibility relation $\mathcal{R}_\mathcal{U}$ nor the partial order $\mathcal{C}_\mathcal{U}$ induced by $(\mathcal{T}_\mathcal{U})$:

Lemma 2 (Existence of Feasible Successor). *If the normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$ of a tree family $(\mathcal{T}_\mathcal{U})$ is acyclic, then for any non-sink node u of $\mathcal{N}_\mathcal{U}$ there exists at least one successor node $v \in \mathcal{N}_\mathcal{U}$ that is a* feasible *successor of u in $\mathcal{G}_\mathcal{U}$, in the sense that $\mathcal{C}_\mathcal{U}(v) \cap \mathcal{R}_\mathcal{U}(u) = \emptyset$ and $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(v) = \emptyset$.*

Proof. Note that $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(u) = \emptyset$ and that the assumption $(u, v) \in \mathcal{A}_\mathcal{U}$ ensures $v \in \mathcal{C}_\mathcal{U}(u)$ and $v \neq u$ and $v \notin \mathcal{C}_\mathcal{U}(v)$ (because the transitive closure is not reflexive). Hence $\mathcal{C}_\mathcal{U}(v) \subset \mathcal{C}_\mathcal{U}(u)$ and as consequence $\mathcal{C}_\mathcal{U}(v) \cap \mathcal{R}_\mathcal{U}(u) \subseteq \mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(u) = \emptyset$.

If u has a single successor v in $\mathcal{N}_\mathcal{U}$, then let us assume $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(v) \neq \emptyset$. Consequently, there exists a node $w \in \mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(v)$. Moreover, the assumption that u has a single successor v ensures that $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{C}_\mathcal{U}(v) = \mathcal{C}_\mathcal{U}(u) \setminus \{v\}$ because the transitive closure is not reflexive. Hence we have $w \in \mathcal{C}_\mathcal{U}(v) \cap \mathcal{R}_\mathcal{U}(v)$, which is impossible: contradiction.

If u has at least two distinct successors v and v' in $\mathcal{N}_\mathcal{U}$, then things are more complicated. Consider Figure 3. Assuming $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(v) \neq \emptyset$ and $\mathcal{C}_\mathcal{U}(u) \cap \mathcal{R}_\mathcal{U}(v') \neq \emptyset$, take $w \in \mathcal{C}_\mathcal{U}(v) \cup \{v\}$ and $w' \in \mathcal{C}_\mathcal{U}(v') \cup \{v'\}$ such that:

$$\exists \ell' \in \Phi_\mathcal{U}^{-1}(v') : w \in \mathcal{R}_\mathcal{U}(\ell') \tag{1}$$

$$\exists \ell \in \Phi_\mathcal{U}^{-1}(v) : w' \in \mathcal{R}_\mathcal{U}(\ell) \tag{2}$$

Property 4 ensures that $w' \in \mathcal{R}_\mathcal{U}(v)$ and $w \in \mathcal{R}_\mathcal{U}(v')$. Now, consider the classification tree $\mathcal{T}_w = (\mathcal{N}_w, \mathcal{A}_w)$ extracted from $\mathcal{G}_\mathcal{U}$ in the following way:

- $\mathcal{N}_w = \{\ell_1, \ell_2, \ell_3, w, r\}$, where $\ell_1 \in \Phi_\mathcal{U}^{-1}(v) \setminus \Phi_\mathcal{U}^{-1}(u)$, $\ell_2 \in \Phi_\mathcal{U}^{-1}(u)$, $\ell_3 \in \Phi_\mathcal{U}^{-1}(v') \setminus \Phi_\mathcal{U}^{-1}(u)$, and r is the single sink of $\mathcal{G}_\mathcal{U}$.
- The relationships between the nodes of $\mathcal{N}_w$ are $\ell_1 \preceq w$, $\ell_2 \preceq w$, $w \preceq r$, $\ell_3 \preceq r$, and $w \in \mathcal{R}_\mathcal{U}(\ell_3)$.

Such an extraction is done by first taking the subgraph of $\mathcal{G}_\mathcal{U}$ induced by $\mathcal{N}_w$ and the descendants of $\mathcal{N}_w$, and then removing the nodes with in-degree 1. Formula (1) ensures that the source ℓ_3 exists. Similarly, let $\mathcal{T}_{w'} = (\mathcal{N}_{w'}, \mathcal{A}_{w'})$ be the classification tree extracted from $\mathcal{G}_\mathcal{U}$ in the following way:

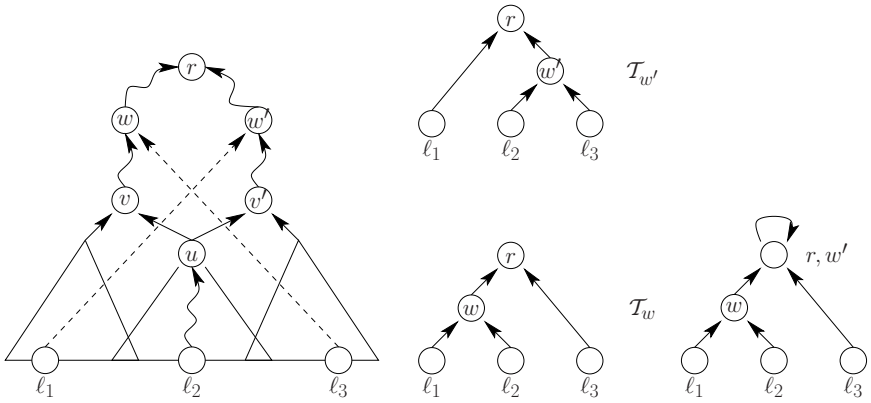
- $\mathcal{N}_{w'} = \{\ell_1, \ell_2, \ell_3, w', r\} = \{w'\} \cup (\mathcal{N}_w \setminus \{w\})$.
- The relationships between the nodes of $\mathcal{N}_{w'}$ are $\ell_2 \preceq w'$, $\ell_3 \preceq w'$, $w' \preceq r$, $\ell_1 \preceq r$, and $w' \in \mathcal{R}_\mathcal{U}(\ell_1)$.

Formula (2) ensures that the source ℓ_3 exists. But the trees $\mathcal{T}_w$ and $\mathcal{T}_{w'}$ have a normal form that contains a circuit (see Figure 3c), and hence $\mathcal{G}_\mathcal{U}$ contains a circuit, so there is a contradiction with the initial assumption that $\mathcal{G}_\mathcal{U}$ is acyclic. □

We can now state the main result of this section:

Theorem 1 (Extractability). *There exists a tree that is compatible with the tree family $(\mathcal{T}_\mathcal{U})$ if and only if there exists an acyclic normal form of $(\mathcal{T}_\mathcal{U})$.*

Proof. First, we show that the condition is necessary, i.e., that if a circuit is detected in $\mathcal{G}_\mathcal{U}$, then there does not exist any tree compatible with the tree family $(\mathcal{T}_\mathcal{U})$. Assume



(a) The normal form $\mathcal{G}_{\mathcal{U}}$.

(b) Two trees $\mathcal{T}_w$ and $\mathcal{T}_{w'}$ extracted from $\mathcal{G}_{\mathcal{U}}$.

(c) Normal form of $\mathcal{T}_w$ and $\mathcal{T}_{w'}$.

Fig. 3: Proof of Lemma 2. Plain arcs represent directed arcs of $\mathcal{A}_{\mathcal{U}}$, dashed arcs represent the impossibility relation, and curly arcs depict paths in $\mathcal{G}_{\mathcal{U}}$.

that such a compatible tree $\mathcal{A}$ exists nevertheless when $\mathcal{G}_{\mathcal{U}}$ contains a circuit. Let $\mathcal{T}_i$ and $\mathcal{T}_j$ be two trees of the family $(\mathcal{T}_{\mathcal{U}})$. By the definition of the normal form (Definition 3), there exist two subgraphs $\mathcal{H}_i$ and $\mathcal{H}_j$ of $\mathcal{G}_{\mathcal{U}}$ such that $\mathcal{H}_i \approx \mathcal{T}_i$ and $\mathcal{H}_j \approx \mathcal{T}_j$. Moreover, assume the graph $\mathcal{H}_i \cup \mathcal{H}_j$ contains a circuit (and thereby $\mathcal{G}_{\mathcal{U}}$ contains a circuit). But then there exist two subtrees $\mathcal{A}_i$ and $\mathcal{A}_j$ of $\mathcal{A}$ such that $\mathcal{A}_i \approx \mathcal{H}_i$ and $\mathcal{A}_j \approx \mathcal{H}_j$. Hence $\mathcal{A}_i \cup \mathcal{A}_j$ contains a circuit and $\mathcal{A}$ cannot be a compatible tree with $(\mathcal{T}_{\mathcal{U}})$ because it contains a circuit. There is a contradiction with the assumption that $\mathcal{G}_{\mathcal{U}}$ contains a circuit.

Second, we show that the condition is sufficient. Assume the normal form $\mathcal{G}_{\mathcal{U}}$ does not contain any circuit. The following procedure extracts a compatible tree from the normal form $\mathcal{G}_{\mathcal{U}}$. Starting from a directed graph only containing the nodes of $\mathcal{G}_{\mathcal{U}}$ but no arcs, a breadth-first traversal of $\mathcal{G}_{\mathcal{U}}$, originating from its sources, assigns a single feasible successor v to the current node u such that $(u, v) \in \mathcal{A}_{\mathcal{U}}$. Such a feasible successor always exists, by Lemma 2. Notice that with the previous procedure a compatible tree can be extracted from the normal form in $O(\ell)$ time.

The extracted digraph is a tree (but not necessarily complete) because each node has a single successor, it does not contain any transitive arcs, and it has a single sink node. Finally, we may have to perform some post-processing, in $O(\ell)$ time, in order to contract all arcs (u, v) where u is the only node with successor v (that is, where v has in-degree 1). □

The time complexity for evaluating the necessary and sufficient condition induced by Theorem 1 is the sum of the time complexities for computing the normal form of $(\mathcal{T}_{\mathcal{U}})$ (i.e., $O(\ell^2)$ by Section 3.2) and checking that $\mathcal{G}_{\mathcal{U}}$ is acyclic (i.e., $O(\ell)$ by a depth-first traversal of $\mathcal{G}_{\mathcal{U}}$), hence $O(\ell^2)$ time.

4 Enumerating All the Compatible Trees

We introduce a constraint programming approach to the classification tree problem. The aim is to propose a framework that allows the enumeration of *all* the trees that are compatible with a given tree family. We model this problem by a new $orderedTree$ global constraint in Section 4.1, and present a filtering algorithm in Section 4.2.

4.1 Modelling the Classification Tree Problem

In constraint programming, a directed tree $\mathcal{T}$ of n non-root nodes is usually modelled by a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ of *successor variables*, where successor variable v_i is associated with non-root node i of $\mathcal{T}$. The domain of successor variable v_i, denoted by $dom(v_i)$, contains the potential successors of node i within $\mathcal{T}$, including its root. In a solution, each successor variable is assigned exactly one value of its domain, thereby enforcing one outgoing arc to each non-root node of $\mathcal{T}$, such that the resulting digraph is acyclic. An arc (i, j) corresponding to successor variable v_i being assigned a value $j \in dom(v_i)$ is called a *required arc*. An arc (i, j) corresponding to a value $j \in dom(v_i)$, when successor variable v_i is not instantiated yet (that is, when $|dom(v_i)| \geq 2$), is called a *possible arc*.

The proposed new constraint has the signature $orderedTree(\mathcal{V}, \mathcal{G}_\mathcal{U}, \mathcal{C}_\mathcal{U}, \mathcal{C}_\mathcal{U}^{-1}, \mathcal{R}_\mathcal{U})$, where $\mathcal{V}$ is the set of successor variables representing a directed tree $\mathcal{T}$ that is compatible with the given tree family $(\mathcal{T}_\mathcal{U})$ of normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$, whose transitive closure is graph $\mathcal{C}_\mathcal{U}$ and its inverse $\mathcal{C}_\mathcal{U}^{-1}$, and where $\mathcal{R}_\mathcal{U}$ is the impossibility relation induced by $(\mathcal{T}_\mathcal{U})$. The digraph $\mathcal{T}$ is initially given by $(\mathcal{S}_\mathcal{U} \cup \mathcal{I}_\mathcal{U}, \mathcal{A})$, where $\mathcal{S}_\mathcal{U}$ is the set of classified elements of $(\mathcal{T}_\mathcal{U})$ (and thus the set of sources of $\mathcal{G}_\mathcal{U}$), while $\mathcal{I}_\mathcal{U}$ is the set of internal nodes of $\mathcal{G}_\mathcal{U}$, and $\mathcal{A} = \{(u, v) \mid u \in \mathcal{S}_\mathcal{U} \cup \mathcal{I}_\mathcal{U} \wedge v \in \mathcal{I}_\mathcal{U}\}$ is the set of possible arcs. In other words, $\mathcal{T}$ is initially a complete graph over its nodes except that there are no arcs between any of its sources. A solution to an $orderedTree$ constraint is defined by:

Definition 4 (Solution to an $orderedTree$ constraint). *A ground instance of an $orderedTree(\mathcal{V}, \mathcal{G}_\mathcal{U}, \mathcal{C}_\mathcal{U}, \mathcal{C}_\mathcal{U}^{-1}, \mathcal{R}_\mathcal{U})$ constraint is a* solution *if and only if the directed tree represented by $\mathcal{V}$ is compatible with the tree family $(\mathcal{T}_\mathcal{U})$ of normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$, whose transitive closure is $\mathcal{C}_\mathcal{U}$, and impossibility relation $\mathcal{R}_\mathcal{U}$.*

Computing the normal form $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$ of a tree family $(\mathcal{T}_\mathcal{U})$ with ℓ distinct sources can be done in $O(\ell^2)$ time (by Section 3.2). Computing the impossibility relation $\mathcal{R}_\mathcal{U}$ induced by $(\mathcal{T}_\mathcal{U})$ and computing the transitive closure $\mathcal{C}_\mathcal{U}$ of $\mathcal{G}_\mathcal{U}$ can be done in $O(\ell^2)$ time, by performing a depth-first traversal from each source node of $\mathcal{G}_\mathcal{U}$.

Let n be the number of non-root nodes of $\mathcal{G}_\mathcal{U}$. (Recall that all sinks of the computed normal form are given the same new successor r before returning the normal form, so that r is its unique sink.) Note that we necessarily have $n = \Theta(\ell)$. We model the wanted directed tree $\mathcal{T}$ by a set $\mathcal{V} = \{v_1, \ldots, v_n\}$ of successor variables. Note that we thus have a number of successor variables (that is, non-root nodes) that is an *upper* bound on the number of non-root nodes needed for any tree $\mathcal{T}$ that is compatible with $(\mathcal{T}_\mathcal{U})$. In consequence, the resulting tree $\mathcal{T}$ is not necessarily complete, unlike required

by Definition 1, and we may have to perform some post-processing, in $O(\ell)$ time, in order to contract all arcs (u, v) where u is the only node with successor v (that is, where v has in-degree 1).

4.2 Filtering Algorithm

We now propose a complete filtering algorithm for the $orderedTree$ constraint. The infeasible values in the domains of the variables of $\mathcal{V}$ according to an $orderedTree$ constraint are characterised in as follows:

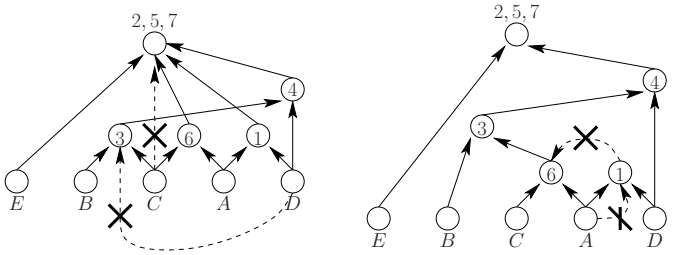
Proposition 1 (Infeasible Values). *For any variable $v_i \in \mathcal{V}$, we have that $j \notin dom(v_i)$ if and only if at least one of the following conditions holds:*

1. $j \in \mathcal{R}_{\mathcal{U}}(i)$
2. $\mathcal{C}_{\mathcal{U}}(j) \cap \mathcal{R}_{\mathcal{U}}(i) \neq \emptyset$
3. $\mathcal{C}_{\mathcal{U}}(i) \cap \mathcal{R}_{\mathcal{U}}(j) \neq \emptyset$
4. (i, j) *is a transitive arc in* $\mathcal{G}_{\mathcal{U}}$ *(that is,* $j \in \mathcal{C}_{\mathcal{U}}(i) \wedge (i, j) \notin \mathcal{A}_{\mathcal{U}}$

Proof. Conditions 1, 2, and 3 directly follow from Lemma 2. Condition 4 is necessary because if a transitive arc (i, j) in $\mathcal{G}_{\mathcal{U}}$ were in a solution to an $orderedTree$ constraint, then the partial order induced by $\mathcal{G}_{\mathcal{U}}$ would not be respected. Moreover, all the conditions are sufficient by considering the procedure described in Section 3.3 based on Lemma 2. □

Consider the example depicted by Figure 4. First, in Figure 4a the arcs $(D, 3)$ and (C, r) are deduced as infeasible by respectively conditions 1 and 4 of Proposition 1. Indeed, the tree $\mathcal{T}_2$ of Figure 1a enforces $3 \in \mathcal{R}_{\mathcal{U}}(D)$, so condition 1 deduces that $(D, 3)$ is infeasible. Since the arc (C, r) is transitive in $\mathcal{G}_{\mathcal{U}}$, condition 4 deduces that (C, r) is infeasible. Second, Figure 4b depicts the current normal form after enforcing the arc $(C, 6)$. The arcs $(1, 6)$ and $(A, 1)$ are then deduced as infeasible by respectively conditions 2 and 3 of Proposition 1. Indeed, we have $\mathcal{C}_{\mathcal{U}}(A) \cap \mathcal{R}_{\mathcal{U}}(1) = \{3\}$ and $\mathcal{C}_{\mathcal{U}}(6) \cap \mathcal{R}_{\mathcal{U}}(1) = \{3\}$.

Algorithm 1 is the proposed filtering algorithm for the $orderedTree$ constraint. The normal form $\mathcal{G}_{\mathcal{U}} = (\mathcal{N}_{\mathcal{U}}, \mathcal{A}_{\mathcal{U}})$, its transitive closure $\mathcal{C}_{\mathcal{U}}$ (and its inverse $\mathcal{C}_{\mathcal{U}}^{-1}$), and the impossibility relation $\mathcal{R}_{\mathcal{U}}$ are updated according to each successor variable that was instantiated since the last awakening of the constraint (lines 2 to 8). This update is done according to the procedures $addArc$ and $updateClosure$. The first one can be done in $O(\ell)$ time because the tests $i \in \mathcal{C}_{\mathcal{U}}(k) \wedge (k, j) \in \mathcal{A}_{\mathcal{U}}$ does not exceed $O(1)$ with a perfect hashing data structure [6], while the second one also takes $O(\ell)$ time. Hence lines 2 to 8 can be done in $O(\ell^2)$ time. Next, if no circuit is detected in the updated normal form (line 9), then the feasibility of each domain value of each variable is ensured according to the conditions of Proposition 1 (lines 10 to 12); otherwise a failure is caught (lines 13 and 14). This can be done in $O(\ell^3)$ time. Overall, Algorithm 1 thus takes $O(\ell^3)$ time.



(a) Condition 1 deduces $(D, 3)$ as infeasible, and condition 4 deduces (C, r), where $r = \{2, 5, 7\}$, as infeasible.

(b) Condition 2 deduces $(1, 6)$ as infeasible, and condition 3 deduces $(A, 1)$ as infeasible.

Fig. 4: Example of filtering induced by the normal form introduced in Figure 1c. Dashed arcs represent some infeasible arcs deduced by Proposition 1.

5 Example of Classification Trees: Evolutionary Trees

One objective of phylogeny is to construct the genealogy of species, called the *tree of life*, whose leaves represent the contemporary species and whose internal nodes represent extinct species that are not necessarily named. An important problem in phylogeny is the construction of a supertree [3] that is compatible with several such evolutionary trees. This notion of evolutionary tree corresponds to our notion of classification tree. Note however that the time line is inverted: a successor in a classification tree actually is an evolutionary predecessor, and the first common descendant (Φ) of a set of leaves in a classification tree actually is referred to in phylogeny as the *most recent common ancestor* ($mrca$) of the species represented by those leaves. The notion of *strong compatibility* [10] corresponds to Definition 1 of tree compatibility. In phylogeny, other notions of compatibility exist: *weak compatibility* [13] and *stable compatibility* [2].

Under strong compatibility, the first supertree algorithm was given in [1], with an application for database management systems; it takes $O(\ell^2 \log \ell)$ time, where ℓ is the number of leaves in the given trees. Derived algorithms have emerged in phylogeny, for instance *OneTree* [10], which takes $O(\ell^2)$ time. The *AllTrees* algorithm [10] in principle constructs *all* compatible trees, but there is in general an exponential (in ℓ) amount of such trees. The first constraint program was proposed in [7] and subsequently improved in [8]. Another constraint program was proposed in [2], but under stable compatibility. Our constraint model has only $\Theta(\ell)$ domain variables, whereas the constraint model of [7, 8] has $\Theta(\ell^2)$ domain variables. Our constraint model and the one of [8] have only one constraint, whereas the constraint model of [7] has $\Theta(\ell^3)$ constraints. Unlike our constraint model, but like the *OneTree* algorithm [10], the constraint models of [7, 8] require a preliminary breaking up of the given trees into so-called triples and fans, in $O(\ell)$ time.

Algorithm 1 Computing the infeasible arcs of $\mathcal{T}$ according to Proposition 1.

procedure $filteringOrderedTree(\mathcal{V}, \mathcal{G}_\mathcal{U}, \mathcal{C}_\mathcal{U}, \mathcal{C}_\mathcal{U}^{-1}, \mathcal{R}_\mathcal{U})$;
for all variable v_i just instantiated to some value j **do** {arc (i, j) was just added to $\mathcal{T}$}
  $addArc(i, j)$;
  $updateClosure(i, j)$;
  $\mathcal{R}_\mathcal{U}(j) \leftarrow \mathcal{R}_\mathcal{U}(j) \cup \mathcal{R}_\mathcal{U}(i)$;
  for all node $k \in \mathcal{N}_\mathcal{U}$ **such that** $(i, k) \in \mathcal{A}_\mathcal{U} \wedge k \neq j$ **do**
    $addArc(j, k)$;
    $updateClosure(j, k)$;
if $\mathcal{G}_\mathcal{U} = (\mathcal{N}_\mathcal{U}, \mathcal{A}_\mathcal{U})$ is acyclic **then**
  for all variable $v_i \in \mathcal{V}$ **do**
    for all value $j \in dom(v_i)$ **such that** $j \in \mathcal{R}_\mathcal{U}(i)$ **or** $\mathcal{C}_\mathcal{U}(j) \cap \mathcal{R}_\mathcal{U}(i) \neq \emptyset$ **or** $\mathcal{C}_\mathcal{U}(i) \cap \mathcal{R}_\mathcal{U}(j) \neq \emptyset$ **or** arc (i, j) is transitive in $\mathcal{G}_\mathcal{U}$ **do** {check Proposition 1 for each value j}
      remove value j from $dom(v_i)$;
else
  catch a failure;

procedure $addArc(i, j)$;
add arc (i, j) to $\mathcal{A}_\mathcal{U}$;
for all node $k \in \mathcal{N}_\mathcal{U}$ **such that** $i \in \mathcal{C}_\mathcal{U}(k) \wedge (k, j) \in \mathcal{A}_\mathcal{U}$ **do** $\{(k, j)$ is transitive$\}$
  remove arc (k, j) from $\mathcal{A}_\mathcal{U}$;
for all node $k' \in \mathcal{N}_\mathcal{U}$ **such that** $k' \in \mathcal{C}_\mathcal{U}(j) \wedge (i, k') \in \mathcal{A}_\mathcal{U}$ **do** $\{(i, k')$ is transitive$\}$
  remove arc (i, k') from $\mathcal{A}_\mathcal{U}$;

procedure $updateClosure(i, j)$;
for all node $k' \in \mathcal{C}_\mathcal{U}^{-1}(i)$ **do**
  $\mathcal{C}_\mathcal{U}(k') \leftarrow \mathcal{C}_\mathcal{U}(k') \cup \{j\} \cup \mathcal{C}_\mathcal{U}(j)$;
for all node $k'' \in \mathcal{C}_\mathcal{U}(j)$ **do**
  $\mathcal{C}_\mathcal{U}^{-1}(k'') \leftarrow \mathcal{C}_\mathcal{U}^{-1}(k'') \cup \{i\} \cup \mathcal{C}_\mathcal{U}^{-1}(i)$;

6 Conclusion

We have presented a novel constraint-programming (CP) approach to the ubiquitous problem of finding a classification tree over a set of classified elements (such as historical artifacts, languages, or biological species) that is compatible with a given family of classification trees over subsets of those classified elements.

Toward this, we have first introduced a natural and compact graph representation of a family of classification trees over ℓ distinct classified elements. We have also shown that if such a *normal form*, computable in $O(\ell^2)$ time, has a circuit, then the related classification tree problem has no solutions. Furthermore, *one* compatible tree can be extracted from the normal form in $O(\ell)$ time. This algorithm thereby matches the performance of the best known ad hoc algorithms, such as *OneTree* [10]. We have then proposed a novel global constraint, based on this normal form, for modelling the classification tree problem, including a complete filtering algorithm that takes $O(\ell^3)$ time and requires $\Theta(\ell)$ decision variables, hence this is probably the first CP approach to the problem that *provably* takes polynomial time. Our previous CP approach [2], intended for a wider purpose, has not been observed to backtrack on phylogenetic instances of

the classification tree problem, but we had no proof thereof. The same seems to hold for another previous CP approach [8], which enhances [7] but still requires $\Theta(\ell^2)$ decision variables and $\Theta(\ell^3)$ constraints.

While the computation of the normal form as an intermediate representation can technically be omitted, we advocate it as an alternative output to the classification tree problem; witness the recent interest in phylogenetic super-networks rather than supertrees [9]. The evolution of species need not happen by binary speciation, nor is genetic recombination excluded, and there is no limitation in our approach to binary trees, while a focus on the non-tree normal form could accommodate recombination.

Specialised $O(\ell^2)$ time algorithms, such as the ones of [1, 10], do not provide the flexibility of a CP approach, as every combination of side constraints (on nested species [12] or absolute and relative ancestral divergence dates [4], say) or the addition of an objective function (such as the min-cut criterion of [11], say, when switching to an optimisation version of the problem) require a new ad hoc algorithm.

Future work includes providing an efficient practical implementation of our approach. Also, the deployment of a dedicated CP framework that allows us to deal with side constraints is a necessary step for a useful decision tool for the classification tree problem.

Acknowledgements

The first author is supported by grant 70644501 of VR, the Swedish Research Council. We are grateful for the constructive discussions with Nicolas Beldiceanu, Sophie Demassey, and Jean-Xavier Rampon.

References

1. A. Aho, Y. Sagiv, T. Szymanski, and J. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal of Computing*, 10(3):405–421, 1981.
2. N. Beldiceanu, P. Flener, and X. Lorca. Combining tree partitioning, precedence, and incomparability constraints. *Constraints*, 2008.
3. O. R. Bininda-Emonds, editor. *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*. Kluwer, 2004.
4. D. Bryant, C. Semple, and M. Steel. Supertree methods for ancestral divergence dates and other applications. In O. R. Bininda-Emonds, editor, *Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life*, volume 3, pages 129–151. Kluwer, 2004.
5. E. Erdem, V. Lifschitz, L. Nakhleh, and D. Ringe. Reconstructing the evolutionary history of Indo-European languages using answer set programming. In V. Dahl and P. Wadler, editors, *Proceedings of PADL'03*, volume 2562 of *LNCS*, pages 160–176. Springer-Verlag, 2003.
6. M. L. Fredman, J. Komlós, and E. Szemerédi. Storing a sparse table with 0(1) worst case access time. *J. ACM*, 31(3):538–544, 1984.
7. I. Gent, P. Prosser, B. Smith, and W. Wei. Supertree construction using constraint programming. In F. Rossi, editor, *Proceedings of CP'03*, volume 2833 of *LNCS*, pages 837–841. Springer-Verlag, 2003.
8. N. Moore. Species trees and the ultrametric constraint. Final-year project report at the Department of Computing Science, University of Glasgow, UK, June 2007.

9. L. Nakhleh and L.-S. Wang. Phylogenetic networks, trees, and clusters. In *Proceedings of IWBRA'05*, volume 3515 of *LNCS*, pages 919–926. Springer-Verlag, 2005.
10. M. P. Ng and N. Wormald. Reconstruction of rooted trees from subtrees. *Discrete Applied Mathematics*, 69:19–31, 1996.
11. R. Page. Modified mincut supertrees. In *Proceedings of WABI'02*, volume 2452 of *LNCS*, pages 537–551. Springer-Verlag, 2002.
12. C. Semple, P. Daniel, W. Hordijk, R. Page, and M. Steel. Supertree algorithms for ancestral divergence dates and nested taxa. *Bioinformatics*, 20:2355–2360, 2004.
13. M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992.