# $\mathrm{FO}^2(<, +1, \sim)$ on data trees, data tree automata and an extension of BVASS

Jérémie Dimino, Florent Jacquemard[1] and Luc Segoufin[2]

[1] jeremie@dimino.org
[2] INRIA & Ircam, Paris, France. florent.jacquemard@inria.fr
[3] INRIA & LSV, Cachan, France. luc.segoufin@inria.fr

**Abstract.** A data tree is a unranked ordered tree where each node carries a label from a finite alphabet and a datum from some infinite domain. We consider the two variable first order logic $\mathrm{FO}^2(<, +1, \sim)$ over data trees. Here $+1$ refers to the child and the next sibling relations while $<$ refers to the descendant and following sibling relations. Moreover $\sim$ is a binary predicate testing data equality. We exhibit an automata model, denoted $\mathrm{DA}^{\#}$, that is more expressive than $\mathrm{FO}^2(<, +1, \sim)$ and such that emptiness of $\mathrm{DA}^{\#}$ and satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ are equivalent as decision problems.
This is proved via a model of counter tree automata, denoted EBVASS, that extends Branching Vector Addition Systems with States (BVASS) with extra features for merging counters. We show that, as decision problems, reachability for EBVASS, satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ and emptiness of $\mathrm{DA}^{\#}$ are equivalent.

## Introduction

A data tree is a unranked ordered tree where each node carries a label from a finite alphabet and a datum from some infinite domain. It is a convenient model of XML documents [4], where data represent attribute values or text contents. The special case of data words has been considered in the realm of program verification, as they are suitable to model the behavior of concurrent, communicating or timed systems, where data can represent e.g. process identifiers or time stamps [1, 6, 7].

Finding decidable logics or automaton models over data trees is an important quest when studying data-driven systems. Problems like deciding wether two properties of XML documents are equivalent usually reduce to the satisfiability problem, i.e. testing whether a formula is satisfied by some data tree or not. One standard formalism to express properties of XML documents is the logic XPath. Although satisfiability of XPath in the presence of data values is undecidable, there are known decidable data-aware fragments [12, 4, 11, 10, 13].

As advocated in [4], the logic $\mathrm{FO}^2(<, +1, \sim)$ can be seen as a relevant fragment of XPath. Here $\mathrm{FO}^2(<, +1, \sim)$ refers to the two-variable fragment of first order logic over unranked ordered data tree, with predicates for the child and the next sibling relations $(+1)$, predicates for the descendant and following sibling

relations ($<$) and a predicate for testing data equality between two nodes ($\sim$). Over data words, $\mathrm{FO}^2(<, +1, \sim)$ is decidable [5] and its fragment $\mathrm{FO}^2(+1, \sim)$, without the descendant and following sibling relations, is decidable over data trees [4]. But the decidability status of $\mathrm{FO}^2(<, +1, \sim)$ over data trees remains open – see the related work section below.

In this paper we introduce a new model of automata, denoted $\mathrm{DA}^{\#}$, and show that this model is expressive enough to capture the expressive power of $\mathrm{FO}^2(<, +1, \sim)$ over data trees. It is an extension from data words to data trees of the Data Automata (DA) model of [5], chosen with care in order to be expressive enough to capture the logic but also not too powerful for not going beyond the computational power of the satisfiability problem of $\mathrm{FO}^2(<, +1, \sim)$.

Our first result (Section 2) shows that languages of data trees definable in $\mathrm{FO}^2(<, +1, \sim)$ are also recognizable by $\mathrm{DA}^{\#}$. Moreover the construction of the automaton from the formula is effective. This implies that the satisfiability problem of $\mathrm{FO}^2(<, +1, \sim)$ reduces to the emptiness problem of $\mathrm{DA}^{\#}$. Even though $\mathrm{DA}^{\#}$ are more expressive than $\mathrm{FO}^2(<, +1, \sim)$, we will see that emptiness of $\mathrm{DA}^{\#}$ and satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ are actually equivalent as decision problems.

To show this we introduce a model of counter automata, denoted EBVASS, working on binary trees (without data values). This model extends BVASS by allowing new features for merging counters. In a BVASS the value of a counter at a node $x$ is the sum of the values of that counter at the children of $x$, plus or minus some constant specified by the transition relation. In EBVASS a constraint can be added enforcing the following behavior at node $x$: the counters at its children are decreased by the same arbitrary number $n$, then the sum is performed as for BVASS, and finally, the resulting counter is increased by $n$.

Our second result (Section 3) shows that the emptiness problem for $\mathrm{DA}^{\#}$ reduces to the reachability problem for EBVASS. Finally we show in Section 4 that the latter problem can be reduced to the satisfiability of $\mathrm{FO}^2(<, +1, \sim)$, closing the loop. Altogether, this implies that showing (un)decidability of any of these problems would show (un)decidability of the three of them.

*Related work.* This paper is an extension to data trees of the work of [5] over data words. In [5] it is shown that, over data words, satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ and reachability of VASS are interreducible. As the latter is known to be decidable, $\mathrm{FO}^2(<, +1, \sim)$ is decidable. The DA model introduced in [5] (running on data words) is more expressive than $\mathrm{FO}^2(<, +1, \sim)$, and the associated emptiness problem is shown to be equivalent to the reachability problem of VASS. However the obvious extensions of DA to data trees are either too weak to capture $\mathrm{FO}^2(<, +1, \sim)$ or too expressive and undecidable (see Section 5). Here we weaken the strongest of these extensions, still capturing the expressive power of $\mathrm{FO}^2(<, +1, \sim)$, but such that the associated emptiness problem is equivalent to satisfiability of $\mathrm{FO}^2(<, +1, \sim)$.

BVASS is a model of counter automata extending VASS and running on trees with a feature adding counters when branching, see [15] for a survey of the various formalisms equivalent to BVASS. In [4] it is shown that reachability of BVASS reduces to satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ over data trees. As the reachability of

BVASS is a long standing open problem, [4] concluded that showing decidability of $\mathrm{FO}^2(<, +1, \sim)$ seems unlikely in the near future. The issue remains wide open. The extension EBVASS allowed us to show the converse direction, i.e. reducing satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ to reachability of EBVASS.

There are many other works introducing automata or logical formalism for data words or data trees. Some of them are shown decidable using counter automata, see for instance [9, 13]. The link between counter automata and data automata is not surprising as the corresponding models only compare data values via equality. Hence they are invariant by permutation of the data domain and therefore, often, it is enough to count the number of data values satisfying some properties instead of knowing their precise values.

## 1 Preliminaries

*Unranked ordered data forests.* We work with finite unranked ordered trees and forests over an alphabet $\mathbb{E}$, defined inductively as follows: for any $a \in \mathbb{E}$, $a$ is a tree. If $\boldsymbol{t}_1, \cdots, \boldsymbol{t}_k$ is a finite non empty sequence of trees then $\boldsymbol{t}_1 + \cdots + \boldsymbol{t}_k$ is a forest. If $\boldsymbol{s}$ is a forest and $a \in \mathbb{E}$, then $a(\boldsymbol{s})$ is a tree. The set of trees and forests over $\mathbb{E}$ are respectively denoted by $\mathrm{Trees}(\mathbb{E})$ and by $\mathrm{Forests}(\mathbb{E})$. A tree is called unary (resp. binary) when every node has at most one (resp. two) children. We use standard terminology for trees and forests defining nodes, roots, leaves, parents, children, ancestors, descendants, following and preceding siblings. For two nodes $x$ and $y$ of a forest, we use $x \prec y$ to denote the fact that $x$ is an ancestor of $y$ or, equivalently that $y$ is a descendant of $x$. Given a forest $\boldsymbol{t} \in \mathrm{Forests}(\mathbb{E})$, and a node $x$ of $\boldsymbol{t}$, we denote by $\boldsymbol{t}(x)$ the label of $x$ in $\boldsymbol{t}$.

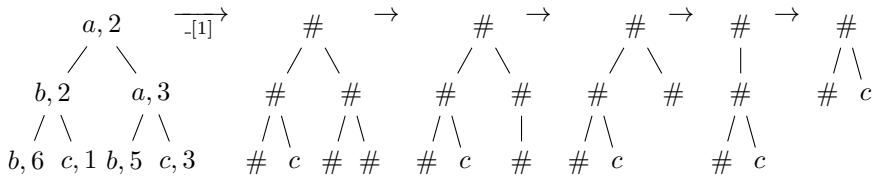
We say that two forests $\boldsymbol{t}_1 \in \mathrm{Forests}(\mathbb{E})$ and $\boldsymbol{t}_2 \in \mathrm{Forests}(\mathbb{F})$ *have the same domain* if there is a bijection from the nodes of $\boldsymbol{t}_1$ to the nodes of $\boldsymbol{t}_2$ that respects the parent and the next-sibling relations. In this case we identify the nodes of $\boldsymbol{t}_1$ with the nodes of $\boldsymbol{t}_2$ and the difference between $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ lies only in the label associated to each node. Given two forests $\boldsymbol{t}_1 \in \mathrm{Forests}(\mathbb{E})$, $\boldsymbol{t}_2 \in \mathrm{Forests}(\mathbb{F})$ having the same domain, we define $\boldsymbol{t}_1 \otimes \boldsymbol{t}_2 \in \mathrm{Forests}(\mathbb{E} \times \mathbb{F})$ as the forest over the same domain but such that for all node $x$, $\boldsymbol{t}_1 \otimes \boldsymbol{t}_2(x) = \langle \boldsymbol{t}_1(x), \boldsymbol{t}_2(x) \rangle$.

The set of **data trees** over a finite alphabet $\mathbb{A}$ and an infinite domain $\mathbb{D}$ is defined as $\mathrm{Trees}(\mathbb{A} \times \mathbb{D})$. Note that every data tree $\boldsymbol{t} \in \mathrm{Trees}(\mathbb{A} \times \mathbb{D})$ can be decomposed into two trees $\boldsymbol{a} \in \mathrm{Trees}(\mathbb{A})$ and $\boldsymbol{d} \in \mathrm{Trees}(\mathbb{D})$ such that $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$.

Let $\mathbb{A}_\#$ be the alphabet which extends $\mathbb{A}$ with a new symbol $\#$. Given a data tree $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ and a data value $d \in \mathbb{D}$, let $\boldsymbol{t}[d] \in \mathrm{Trees}(\mathbb{A}_\#)$ be the tree having the same domain as $\boldsymbol{t}$ with $\boldsymbol{t}[d](x) = \boldsymbol{a}(x)$ if $\boldsymbol{d}(x) = d$ and $\boldsymbol{t}[d](x) = \#$ otherwise. Given $\boldsymbol{t} \in \mathrm{Trees}(\mathbb{A}_\#)$ we denote by $\boldsymbol{t}{\downarrow_\#}$ the tree defined from $\boldsymbol{t}$ by applying the following confluent rewriting rules at any node of $\boldsymbol{t}$ until none of them apply (where $x$ ranges over $\mathrm{Forests}(\mathbb{A}_\#)$).

$$\#(\#(x)) \to \#(x), \quad \#(x) + \# \to \#(x), \quad \# + \#(x) \to \#(x)$$

*Example 1.* Applying the above transformations to the following data tree gives this reduction sequence.




*Logics on data forests.* Data forests are seen as models for first order logic. The domain of such a model is the set of nodes in the forest. Let $\mathrm{FO}^2(<, +1, \sim)$ be the set of first order sentences with two variables built with the predicates $a(x)$ ($a \in \mathbb{A}$ is the label of $x$), $x \sim y$ ($x$ and $y$ carry the same data value), $x = y$ (the nodes are equal), $E_{\rightarrow}(x, y)$ ($y$ is the sibling immediately next to $x$), $E_{\downarrow}(x, y)$ ($x$ is the parent of $y$), and $E_{\rightrightarrows}$, $E_{\downdownarrows}$ which are the non reflexive transitive closures respectively of $E_{\rightarrow}$ and $E_{\downarrow}$, minus respectively $E_{\rightarrow}$ and $E_{\downarrow}$ (i.e. they define two or more navigation steps). The reason for this non standard definition of $E_{\rightrightarrows}$ and $E_{\downdownarrows}$ is that it will be convenient that equality, $E_{\rightarrow}$, $E_{\downarrow}$, $E_{\rightrightarrows}$ and $E_{\downdownarrows}$ are disjoint binary relations.

We will often make use of the macro, $x \parallel y$, and say that $x$ *is parallel to* $y$, as a shortcut for the formula $x \neq y \wedge \neg E_{\rightarrow}(x, y) \wedge \neg E_{\rightarrow}(y, x) \wedge \neg E_{\downarrow}(x, y) \wedge \neg E_{\downarrow}(y, x) \wedge \neg E_{\rightrightarrows}(x, y) \wedge \neg E_{\rightrightarrows}(y, x) \wedge \neg E_{\downdownarrows}(x, y) \wedge \neg E_{\downdownarrows}(y, x)$

We also consider the extension $\mathrm{MSO}^2(<, +1, \sim)$ of $\mathrm{FO}^2(<, +1, \sim)$ with existentially quantified monadic second order variables. Every formula of $\mathrm{MSO}^2(<,+1,\sim)$ has the form $\exists R_1 \ldots \exists R_n\ \phi$ where $\phi$ is a $\mathrm{FO}^2(<, +1, \sim)$ formula called the *core*, involving the variables $R_1, \ldots, R_n$ as unary predicates.

*Automata model for data forests.* We now define the automata model, denoted $\mathrm{DA}^{\#}$, that we use in this paper. A $\mathrm{DA}^{\#}$ is a pair $(A, B)$ where $A$ is a nondeterministic letter-to-letter transducer taking as input a forest in Forests($\mathbb{A}$) and returning a forest in Forests($\mathbb{B}$) with the same domain, while $B$ is a forest automata taking as input trees in Forests($\mathbb{B}$).

A data forest $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d} \in \mathrm{Forests}(\mathbb{A} \times \mathbb{D})$ is accepted by $(A, B)$ if

1. there exists a forest $\boldsymbol{a}' \in \mathrm{Forests}(\mathbb{B})$ such that $\boldsymbol{a}'$ is a possible output of $A$ on $\boldsymbol{a}$ and,
2. for all $d \in \mathbb{D}$, $(\boldsymbol{a}' \otimes \boldsymbol{d})[d]{\downarrow_{\#}}$ is accepted by $B$.

Hence, $\mathrm{DA}^{\#}$ is an generalization to forests of the data automata model of [5], with a normalization using the rewriting rules before running $B$. This normalization step is necessary in order to avoid undecidability as shown in Section 5.

In the following straightforward lemma, we use the term letter projection for a relabeling function defined as $h : \mathbb{A}' \to \mathbb{A}$, where $\mathbb{A}$ and $\mathbb{A}'$ are alphabets.

**Lemma 1.** *The class of* $\mathrm{DA}^{\#}$ *languages is closed under union, intersection and letter projection.*

## 2 From $\mathrm{FO}^2(<, +1, \sim)$ to $\mathrm{DA}^{\#}$

In this section we show the following result.

**Theorem 1.** *Given a formula $\phi$ in $\mathrm{FO}^2(<, +1, \sim)$, there exists a $\mathrm{DA}^{\#}$ $D$, effectively computable from $\phi$, accepting the set of data forests satisfying $\phi$.*

The proof works in two steps. During the first step we provide a normal form for sentences of $\mathrm{FO}^2(<, +1, \sim)$ that is essentially an $\mathrm{MSO}^2(<, +1, \sim)$ formula whose core is a conjunction of simple formula of $\mathrm{FO}^2(<, +1, \sim)$. In a second step we show that each of the conjunct can be translated into a $\mathrm{DA}^{\#}$, and we conclude using composition of these automata by conjunction, see Lemma 1.

*Intermediate Normal Form.* We show that every $\mathrm{FO}^2(<, +1, \sim)$ formula can be transformed into an equivalent $\mathrm{MSO}^2(<, +1, \sim)$ formula in *intermediate normal form*:

$$\exists S_1 \cdots \exists S_k \bigwedge_i \chi_i$$

where each $\chi_i$ has one of the two following simple forms:

$$\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge \delta(x, y) \to \gamma(x, y) \tag{1}$$

$$\forall x \exists y\ \alpha(x) \to (\beta(y) \wedge \delta(x, y) \wedge \epsilon(x, y)) \tag{2}$$

where each of $\alpha$ and $\beta$ is a *type*: a conjunction of monadic predicates or their negation (these unary predicates are either from $\mathbb{A}$ or from $S_1, \ldots, S_k$, *i.e.* introduced by the existentially quantified variables), $\delta(x, y)$ is either $x \sim y$ or $x \not\sim y$, $\gamma(x, y)$ is one of $\neg E_{\rightrightarrows}(x, y)$, $\neg E_{\downdownarrows}(x, y)$ or $\neg(x \parallel y)$, and $\epsilon(x, y)$ is one of $x = y$, $E_{\to}(x, y)$, $E_{\to}(y, x)$, $E_{\downarrow}(x, y)$, $E_{\downarrow}(y, x)$, $E_{\rightrightarrows}(x, y)$, $E_{\rightrightarrows}(y, x)$, $E_{\downdownarrows}(x, y)$, $E_{\downdownarrows}(y, x)$, $x \parallel y$ or *false*.

This normal form is obtained by simple syntactical manipulation very similar to the one given in [5] for the data words case. The proof is omitted in the extended abstract (see Appendix A for the details).

*Construction of the Automaton.* We now show how to transform a formula in intermediate normal form into a $\mathrm{DA}^{\#}$. Let $\mathbb{A}$ be the initial alphabet and let $\mathbb{A}'$ be the new alphabet formed by combining letters of $\mathbb{A}$ with the newly quantified monadic predicates $S_1, \ldots, S_k$. By closure of $\mathrm{DA}^{\#}$ under intersection and letter projection (Lemma 1), it is enough to construct a $\mathrm{DA}^{\#}$ automaton for each formula of the form (1) or (2), accepting the data trees in $\mathrm{Trees}(\mathbb{A}' \times \mathbb{D})$ satisfying the formula. In the discussion below, a node whose labels satisfies the type $\alpha$ will be called an *$\alpha$-node*.

We do a case analysis depending on the atoms involved in the formula of the form (1) or (2). For each case we construct a $\mathrm{DA}^{\#}$ $(A, B)$ recognizing the set of data trees satisfying the formula. The construction borrows several ideas from the data word case [5], but some extra work is needed as the tree structure is more complicated. Many of the cases build on generic constructions that we describe in the following remark.

*Remark 1.* It will be useful for the DA$^{\#}$ $(A, B)$ to distinguish a finite number of specific data values. We will then say that $(A, B)$ *marks the data value of a node* $x$ *using the new color* $c$. This can be done as follows. The transducer $A$ marks the node $x$ with a specific new color $c'$. At the same time it guesses all the nodes sharing the same data value as $x$ and marks them with a new color $c$. Then, the forest automaton $B$ checks, for every data value, that either none of the nodes are marked with $c$ or $c'$, or that all nodes not labeled with # are marked with $c$ or $c'$ and that $c'$ occurs exactly once in the same tree. It is now clear that for the run to be accepting, $A$ must color exactly one data value and that all the nodes carrying this data value must be marked with $c$ and $c'$. The transducer $A$ can then build on this fact for checking other properties. A generic example of the usefulness of this behavior is given below.

Once a data value is marked with a color $c$, then a property of the form $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \gamma(x, y)$ is a conjunction of $\forall x \forall y\ \alpha(x) \wedge c(x) \wedge \beta(y) \wedge \neg c(y) \rightarrow \gamma(x, y)$ with $\forall x \forall y\ \alpha(x) \wedge \neg c(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \gamma(x, y)$. The first part, $\forall x \forall y\ \alpha(x) \wedge c(x) \wedge \beta(y) \wedge \neg c(y) \rightarrow \gamma(x, y)$ is now a regular property and can therefore be tested by $A$. Hence it is enough to consider the case where $x$ does not carry the marked data value. The same reasoning holds if two data values are marked or if the formula starts with a $\forall x \exists y$ quantification.

The complete case analysis is omitted in this extended abstract (see Appendix B). We only provide below one of the most difficult cases, where the formula has the form $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \neg(x \parallel y)$. It expresses the property that every two nodes of type respectively $\alpha$ and $\beta$ and with different data values cannot be parallel (recall that parallel means not parents and not siblings).

*Subcase 1:* There exists two $\alpha$-nodes that are parallel.

Let $x_1$ and $x_2$ be two parallel $\alpha$-nodes and let $z$ be their least common ancestor. We can choose $x_1$ and $x_2$ such that none of the $\alpha$-nodes are parallel to $z$ or sibling of $z$, because if this was not the case then there is an $\alpha$-node $x_3$ parallel to $z$ or sibling of $z$, and therefore $x_3$ is parallel to $x_1$, and we can replace $x_2$ with $x_3$, continuing with a strictly higher node $z$. Let $z_1$ and $z_2$ be the children of $z$ that are respectively ancestors of $x_1$ and $x_2$. Note that by construction, $z_1 \neq z_2$. If $x_1 = z_1$ and there is an $\alpha$-node $x_3$ in the subtree of $z$, different from $x_1$ and parallel to $z_2$, then we replace $x_1$ by $x_3$ and proceed. In other words we ensure that if $x_1 = z_1$ then there is no $\alpha$-node parallel to $z_2$ in the subtree of $z$. We do the same trick to enforce that if $x_2 = z_2$ then there is no $\alpha$-node parallel to $z_1$ in the subtree of $z$. Notice that we cannot have at the same time $x_1 = z_1$ and $x_2 = z_2$. All these properties can be specified in MSO and therefore can be tested by a tree automaton. Let $d_1$ and $d_2$ be the data values of $x_1$ and $x_2$ (possibly $d_1 = d_2$).

Consider now a $\beta$-node $y$ whose data value is neither $d_1$ nor $d_2$. If $y$ is parallel to $z$ or sibling of $z$, then the formula cannot be true as it is contradicted by $(x_1, y)$. If $y$ is an ancestor of $z$ then, as no $\alpha$-node is parallel to $z$, none is parallel to $y$, and hence the formula cannot be falsified with this $y$. Assume now that $y$ is inside the subtree of $z$. If $y = z_1$ and $x_2 \neq z_2$, then the formula is contradicted by

$(x_2, y)$. If $y = z_1$ and $x_2 = z_2$, then, by hypothesis, there is no $\alpha$-node parallel to $y$ in the subtree of $z$, and there is no $\alpha$-node parallel to $y$ outside the subtree of $z$, and altogether, and the formula holds for $y$. If $y \neq z_1$ and $y$ is a descendant of $z_1$, then the formula is contradicted by $(x_2, y)$. The cases where $y$ is descendant of $y_2$ are symmetric: in this case, the formula can only be true if $y = y_2$ and $x_1 = z_1$. The remaining cases, where $y$ is in the subtree of $z$ and not in the subtrees of $y_1$ and $y_2$, all make the formula false. Indeed, in each of these cases, either $(x_1, y)$ or $(x_2, y)$ contradicts the formula.

With this discussion in mind, this case can be solved as follows: The transducer $A$ guesses the nodes $x_1$, $x_2$, $z_1$, $z_2$ and $z$ and checks that they satisfy the appropriate properties. Moreover, $A$ guesses whether $d_1 = d_2$ and marks accordingly the data values of $x_1$ and $x_2$ with one or two new colors. The forest automaton $B$ will then check that the data values are marked appropriately as in Remark 1.

Moreover $A$ checks that for all marked $\beta$-nodes there is no $\alpha$-node parallel to it and with a different data value, a regular property as explained in Remark 1. It now remains for $A$ to check that every unmarked $\beta$-node $y$ behaves according to the discussion above: $y$ is an ancestor of $z$ or $y = z_1$ and $x_2 = z_2$ or $y = z_2$ and $x_1 = z_1$. This is a regular property testable by $A$.

*Subcase 2:* There are no two parallel $\alpha$-nodes.

Let $x$ be an $\alpha$-node such that no $\alpha$-node is a descendant of $x$. By hypothesis, all $\alpha$-nodes are either ancestors or siblings of $x$. Let $d$ be the data value of $x$. We distinguish between several subcases depending on whether there are other $\alpha$-nodes that are siblings of $x$ or not.

If there is an $\alpha$-node $x'$ that is a sibling of $x$, then let $d'$ be its data value (possibly $d = d'$). Consider now a $\beta$-node $y$ whose data value is neither $d$ nor $d'$. Then in order to make the formula true, $y$ must be an ancestor or a sibling of $x$.

In this first case, the transducer $A$ guesses the nodes $x$ and $x'$ and marks the corresponding data values with one or two new colors (according to whether $d = d'$ or not). The forest automaton $B$ will then check that the data values are marked correctly as explained in Remark 1. For the marked $\beta$-nodes, the property is regular and can also be checked by $A$. It remains for $A$ to check that every unmarked $\beta$-nodes is either an ancestor of $x$ or a sibling of $x$.

Now, if there are no $\alpha$-nodes that are sibling of $x$, and $y$ is a $\beta$-node whose data value is not $d$, then in order to make the formal true, $y$ cannot be parallel to $x$, and therefore, $y$ can be an ancestor, a descendant or a sibling of $x$.

In this second case, the transducer $A$ guesses the node $x$, marks its data value using a new color. The forest automaton $B$ will then check that the data values were marked correctly as explained in Remark 1. The transducer $A$ checks that all marked $\beta$-nodes make the formula true, and that all unmarked $\beta$-nodes are not parallel to $x$.

## 3 From DA$^{\#}$ to Counter Tree Automata

In this section we show that the emptiness problem of DA$^{\#}$ can be reduced to the reachability of a counter tree automata model that extends the branching vector addition systems (BVASS). Our counter automata model, called EBVASS, runs on binary trees over a finite alphabet.

We aim at showing that, for all DA$^{\#}$ $D$, the set of trees $\boldsymbol{a}$ such that there is a tree $\boldsymbol{d}$ such that $\boldsymbol{a} \otimes \boldsymbol{d}$ is accepted by $D$ can be recognized by some EBVASS.

We will actually only show this for DA$^{\#}$ running over binary data trees. It is well known that unranked trees can be transformed into binary tree using the first-child/right-sibling encoding and that this point of view preserves regularity. Using a simple extension to data trees of this encoding, it is straightforward to show that the general case reduces to the binary data tree case (see Appendix C).

**Lemma 2.** *Given a* DA$^{\#}$ $D$ *there exists a* DA$^{\#}$ $D'$*, effectively computable from* $D$*, accepting exactly the binary encodings of the data trees accepted by* $D$*.*

*Definition of* EBVASS. An EBVASS is a tree automata equipped with counters. It runs over binary trees over a finite alphabet $\mathbb{A}$. It can increase or decrease its counters but cannot perform a zero test. For BVASS, when going up in the tree, the new value of each counter is the sum of its values at the left and right child. An EBVASS can change this behavior using simple arithmetical constraints.

More precisely an EBVASS is a tuple $(\mathbb{A}, Q, q_0, k, \delta)$ where $\mathbb{A}$ is a finite alphabet, $Q$ is a finite set of states, $q_0 \in Q$ is the initial state and $k \in \mathbb{N}$ is the number of counters, and $\delta$ is the set of transitions which are of two kinds: $\epsilon$-transitions (subset denoted $\delta_\epsilon$) and up-transitions (subset denoted $\delta_{\mathrm{u}}$).

Informally, an $\epsilon$-transition may change the current state and increment or decrement one of the counters. Formally, $\delta_\epsilon$ is a finite subset of $(Q \times \mathbb{A}) \times (Q \times U)$ where $U = \{I_i, D_i \ : \ 1 \leq i \leq k\}$ is the set of possible counter updates: $D_i$ stands for *decrement counter* $i$ and $I_i$ stands for *increment counter* $i$. We view each element of $U$ as a vector over $\{-1, 0, 1\}^k$ with only one non-zero position.

Informally, an up-transition depends on the label of the current node and, when the current node is an inner node, on the states reached at its left and right child. It defines a new state and the new value of each counter is the sum of the values of the corresponding counters of the children. Moreover an up-transition can be associated with a set of constraints of the form $C_{i_1} \ominus C_{i_2} \to C_i$ (for inner nodes) which modify this process as follows: before performing the addition of the counters a number $n$ is guessed, the counter $i_1$ of the left child is decreased by $n$, the counter $i_2$ of the right child is decreased by $n$ and, once the addition of the counters has been executed, the counter $i$ is increased by this same number $n$. Note that $n$ must be so that all intermediate values remain positive.

More formally $\delta_{\mathrm{u}}$ is a finite subset of : $(Q \times \mathbb{A} \times Q) \times (Q \times 2^{\chi})$ where $\chi$ is the finite set of elements of the form $C_{i_1} \ominus C_{i_2} \to C_i$ with $1 \leq i_1, i_2, i \leq k$.

A *configuration* of an EBVASS is a pair $(q, f)$ where $q \in Q$ and $f$ is a valuation of the counters, a function from $[k]$ to $\mathbb{N}$. The initial configuration is $(q_0, f_0)$ where $f_0$ is the function setting all counters to 0. There is an *$\epsilon$-transition*

of label $a$ from $(q, f)$ to $(p, g)$ if $(q, a, p, u) \in \delta_\epsilon$ and $g = f + u$ (in particular this implies that $f + u \geq 0$). We write $(q, f) \xrightarrow{a}_\epsilon (p, g)$, if $(p, g)$ can be reached from $(q, f)$ via a finite sequence of $\epsilon$-transitions of label $a$.

Given a binary tree $\boldsymbol{a} \in \text{Trees}(\mathbb{A})$, a *run* $\rho$ of a EBVASS is a function from nodes of $\boldsymbol{a}$ to configurations verifying for all leaf $x$, $\rho(x) = (q_0, f_0)$ and for all nodes $x, x_1, x_2$ of $\boldsymbol{a}$ with $x_1$ and $x_2$ the left and right child of $x$, and $\rho(x) = (p, f), \rho(x_1) = (q_1, f_1), \rho(x_2) = (q_2, f_2)$ there exists $(p_1, g_1), (p_2, g_2)$ such that:

1. $(q_1, f_1) \xrightarrow{\boldsymbol{a}(x_1)}_\epsilon (p_1, g_1)$, $(q_2, f_2) \xrightarrow{\boldsymbol{a}(x_2)}_\epsilon (p_2, g_2)$,
2. $(p_1, \boldsymbol{a}(x), p_2, p, \Theta) \in \delta_{\text{u}}$,
3. for each constraint $\theta \in \Theta$ of the form $C_{i_1} \ominus C_{i_2} \to C_i$ there is a number $n_\theta$ ($n_\theta$ may be 0) and vectors $v_{\theta,1}, v_{\theta,2}, u_\theta \in \mathbb{N}^k$ having $n_\theta$ respectively at position $i_1, i_2$ and $i$ and all other positions set to zero,
4. $h_1 = g_1 - \Sigma_{\theta \in \Theta} v_{\theta,1} \geq 0$ and $h_2 = g_2 - \Sigma_{\theta \in \Theta} v_{\theta,2} \geq 0$, $f = h_1 + h_2 + \Sigma_{\theta \in \Theta} u_\theta$.

The *reachability* problem for an EBVASS asks on input $p \in Q$, whether there is a tree and a run on that tree reaching the configuration $(p, f_0)$ at its root. Without the constraints $\Theta$ we have the usual notion of BVASS [15].

*Remark:* We shall consider in the following some new constraints of the form $\theta = C_{p_1} \ominus C_{p_2} \to_1 C_p$. Such a constraint is interpreted like $C_{p_1} \ominus C_{p_2} \to C_p$, except that the number $n_\theta$ is forced to be equal to 1. They can easily be simulated using $\epsilon$-transitions and auxiliary states, decreasing the counter $C_{p_1}$ at the left child and the counter $C_{p_2}$ at the right child, then performing the up-transition with the remaining constraints and finally increasing the counter $C_p$. Similarly we shall use constraints of the form $\theta = C_{p_1} \ominus \to_1 C_p$ and $\theta = \ominus C_{p_2} \to_1 C_p$ where only the counter at the left (resp. right) children is decreased. Note that these constraints are evaluated *before* the other constraints of the form $C_{p_1} \ominus C_{p_2} \to C_p$.

*Reduction from* DA$^\#$ *to* EBVASS.

**Theorem 2.** *Given a* DA$^\#$ *$D$, there exists a* EBVASS *$E$, effectively computable from $D$, accepting exactly the trees $\boldsymbol{a}$ such that $\boldsymbol{a} \otimes \boldsymbol{d}$ is accepted by $D$ for some $\boldsymbol{d}$.*

*Proof.* Assume $D = (A, B)$ is a data automata running over $\text{Trees}(\mathbb{A} \times \mathbb{D})$, with $A = (Q_A, \mathbb{A}, \mathbb{B}, F_A, \Delta_A)$ and $B = (Q_B, \mathbb{B}, F_B, \Delta_B)$. To simplify the notations, for any tree $\boldsymbol{a} \in \text{Trees}(\mathbb{A})$ we shall write $\boldsymbol{a}' \in A(\boldsymbol{a})$ if the tree $\boldsymbol{a}' \in \text{Trees}(\mathbb{B})$ is a possible output of $A$ on $\boldsymbol{a}$. The forest automaton $B$ is assumed deterministic and complete, *i.e.* for every $\boldsymbol{a}' \in \text{Trees}(\mathbb{B} \cup \{\#\})$, $B$ evaluates into exactly one state of $Q_B$. In particular, let $p_\# \in Q_B$ be the state on which $B$ evaluates on the tree with a single node labeled with #. Moreover we assume *wlog* that the states of $B$ indicate whether the last letter read was a # or not. In the former case we say that the state is a #-state.

To any data tree $\boldsymbol{t} \in \text{Trees}(\mathbb{B} \times \mathbb{D})$, and any data value $d$ occurring in $\boldsymbol{t}$, the state corresponding to the evaluation of $B$ on $\boldsymbol{t}[d]{\downarrow_\#}$ is called *the $B$-state associated to $d$ in $\boldsymbol{t}$*. To any data tree $\boldsymbol{t} \in \text{Trees}(\mathbb{B} \times \mathbb{D})$, we associate a function $K_{\boldsymbol{t}}$

from $Q_B \backslash \{p_\#\}$ into $\mathbb{N}$ such that for all $p \in Q_B \backslash \{p_\#\}$, $K_{\boldsymbol{t}}(p)$ is the number of data values having $p$ as associated $B$-state on $\boldsymbol{t}$. Note that because $B$ is deterministic and complete, there is at most one non #-state $p$ such that $K_{\boldsymbol{t}}(p) \neq 0$ and actually $K_{\boldsymbol{t}}(p) = 1$. This state is the one associated to the data value of the root of $\boldsymbol{t}$. We call this special state the *root-state* of $\boldsymbol{t}$.

We now construct an EBVASS $E = (\mathbb{A}, Q, q_0, k, \delta)$ with $k = |Q_B| - 1$ recognizing the projection on $\mathbb{A}$ of the data trees accepted by $D$. Each configuration of $E$ is of the form $(q, f)$ where $q \in Q$ and $f$ is a function associating to each counter its value. In the remaining part of this proof we view each such function $f$ as a function from $Q_B$ into the natural numbers. The EBVASS $E$ is constructed such that every state of $Q$ is made of a state of $Q_A$, a state of $Q_B$, together with auxiliary information. For each $q \in Q$ we call the state of $Q_A$ contained in $q$ the $A$-state of $q$. $E$ will also ensure the following property:

($\diamond$) if $E$ can reach the configuration $(q, f)$ at a non leaf mode, then $f$ has only one non #-state $p$ verifying $f(p) \neq 0$ and actually $f(p) = 1$.

We will refer to this state $p$ as the *root-state* of $f$. The EBVASS $E$ is constructed such that every state $q \in Q$ contains the root-state of the current configuration, denoted as the *root-state* of $q$ in the sequel. Finally $E$ will verify the following property which clearly implies the result:

($\star$) $E$ reaches the configuration $(q, f)$ at the root of a tree $\boldsymbol{a}$ iff there is a data tree $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ and $\boldsymbol{a}' \in A(\boldsymbol{a})$ such that $K_{\boldsymbol{a}' \otimes \boldsymbol{d}} = f|_{Q_B \backslash \{p_\#\}}$ and there is a run of $A$ witnessing $\boldsymbol{a}' \in A(\boldsymbol{a})$ whose value at the root of $\boldsymbol{a}$ is the $A$-state of $q$.

Notice that the property ($\star$) is invariant under permutations of $\mathbb{D}$. Hence if a tree $\boldsymbol{d}$ witnesses the property ($\star$), then any tree $\boldsymbol{d}'$ constructed from $\boldsymbol{d}$ by permuting the data values is also a witness for ($\star$). This observation will be useful for showing the correctness of the construction of $E$.

Intuitively $E$ works as follows. Let $x$ be a node of $\boldsymbol{t}$ whose label is $a \in \mathbb{A}$. From configurations $(q_1, f_1)$ and $(q_2, f_2)$ reached respectively at the left and right children $x_1$ and $x_2$ of $x$, $E$ does the following: it simulates $A$ using $a$ and the $A$-states of $q_1$ and $q_2$ obtaining the new $A$-state for $x$ together with the relabeling $a' \in \mathbb{B}$ of $x$. Now $E$ needs to simulate $B$ on all data values, updating the counters appropriately. In order to do this, $E$ has to "guess" the data value $d$ of $\boldsymbol{d}$ at $x$, assuming that it has already guessed the subtrees $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ of $\boldsymbol{d}$ at $x_1$ and $x_2$. Of course, $E$ cannot make a guess amongst an infinite set of choices, but it is enough for $E$ to guess whether the data value is a new data value or not, *i.e.* whether $d$ is present in $\boldsymbol{d}_1$ or $\boldsymbol{d}_2$. In the case $d$ is already present, $E$ guesses the $B$-state associated to $d$ in the subtrees $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ of $\boldsymbol{t}$ rooted at $x_1$ and $x_2$. Note that because $B$ is deterministic and complete, these guesses imply whether $d$ is the data value of $x_1$ or or/and of $x_2$ in $\boldsymbol{d}$. Moreover $E$ guesses whether the data value at $x_1$ occurs in the subtree $\boldsymbol{d}_2$ at $x_2$ or not, and vice-versa for the data value at $x_2$. These tests make finitely many guesses and we will see that this is enough for $E$ to recover the computations on trees $(\boldsymbol{a}' \otimes \boldsymbol{d})[e]{\downarrow_\#}$ for all data values $e$.

We now work out the details. We set $Q = Q_A \times Q_B \times Q_0$, where $Q_0$ is a finite set of auxiliary control states. For any state $q \in Q$, its first component gives the $A$-state of $q$ while the second provides its root-state.
Let us now define the transitions of $E$. The simulation of $A$ is straightforward: We ensure that for every $\epsilon$-transition $(q, a, q', u)$ of $E$, the $A$-states of $q$ and $q'$ coincide, and that for every up-transition $(q_1, a, q_2, q, \Theta)$ of $E$, there exists a transition of $A$ of the form $(p_1, a, p_2, p, a')$, for some $a' \in \mathbb{B}$ such that $p_1, p_2$ and $p$ are the $A$-states of $q_1$, $q_2$ and $q$.

We now define how the counters are modified during the transitions in order to maintain the properties $(\diamond)$ and $(\star)$. For each of the guesses that $E$ made about how the data values at $x$, $x_1$ and $x_2$ are related, we define a set of constraints $\Theta$ and add the up-transitions $(q_1, a, q_2, q, \Theta)$ into $E$, where $q_1, a, q_2$ and $q$ are related as above in order to simulate $A$. We let $a'$ be the letter guessed by $A$ for relabeling the current node $x$ and let $p_1$ and $p_2$ be the respective root-state of $q_1$ and $q_2$.
Due to space limitations we only provide one case of guess. The others are treated similarly (see Appendix D).

For the case where $E$ *guessed that the data value $d$ of the current node is equal to the data value of both its children*, we set $\Theta$ as the union of the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$ for the unique state $p$ such that $(p_1, a', p_2) \to p$ is a transition of $B$, and constraints of the form $C_{v_1} \ominus C_{v_2} \to C_v$ for each transition $(v_1, \#, v_2) \to v$ of $B$ where $v_1$, $v_2$ are $\#$-states.

Correctness is shown by induction on the depth of the tree based on the inductive hypotheses $(\diamond)$ and $(\star)$. Immediately after defining a set $\Theta$ and adding a transition of the form $(q_1, a, q_2, q, \Theta)$ into the set of transitions of $E$, we show that using this transition maintains $(\diamond)$ and $(\star)$. We assume that $E$ reached the configuration $(q, f)$ at the root $x$ of a tree $\boldsymbol{a}$.

If $x$ is a leaf node, then by definition of EBVASS, $q$ is the initial state of $E$ and $f = f_0$ (the function setting all counters to 0), hence $(\star)$ holds.

If $x$ is an inner node, then let $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ be the subtrees of $\boldsymbol{a}$ rooted at the children $x_1$ and $x_2$ of $x$. By induction on $(\star)$ we have trees $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ such that there is a run of $D$ on $\boldsymbol{t}_1 = \boldsymbol{a}_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{t}_2 = \boldsymbol{a}_2 \otimes \boldsymbol{d}_2$ with the appropriate properties. We construct from $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ and the transition of $E$ giving $(q, f)$ a tree $\boldsymbol{d} = d(\boldsymbol{d}_1, \boldsymbol{d}_2)$ such that $D$ also has the expected run on $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$.

For each constraint $\theta$ of the form $C_{u_1} \ominus C_{u_2} \to C_u$ or $C_{u_1} \ominus C_{u_2} \to_1 C_u$, where $u_1$, $u_2$ $u \in {Q_B}^4$, we let $n_\theta$ be the number used by the run of $E$ when using this transition. By induction hypothesis $(\star)$ there are at least $n_\theta$ data values in $\boldsymbol{d}_1$ having $u_1$ as associated $B$-state in $\boldsymbol{a}_1 \otimes \boldsymbol{d}_1$, and $n_\theta$ data values in $\boldsymbol{d}_2$ having $u_2$ as associated $B$-state in $\boldsymbol{a}_2 \otimes \boldsymbol{d}_2$. For showing $(\star)$, we pick $n_\theta$ such data values in each subtrees and call them the data values associated to $\theta$. We do this for all constraints $\theta$ and we choose the associated data values such that they are all distinct. This is possible because of the semantic of the constraints, making sure the counters are big enough and therefore, by $(\star)$, that there is enough data

[4] For the sake of readability, we index the $C$'s with states of $B$ and not integers.

values. The constraints of the form $C_{u_1} \ominus C_{u_2} \to_1 C_u$ where $u$ is a non #-state, for which $n_\theta = 1$, will ensure the propagation of ($\diamond$). We now apply to $\boldsymbol{d}_2$ a permutation on the data values such that for all $\theta$ the data values associated to $\theta$ in $\boldsymbol{d}_2$ are identified with the ones for $\boldsymbol{d}_1$ and such that all other data values are distinct. In order to simplify the notations we call the resulting tree also $\boldsymbol{d}_2$. We then construct $\boldsymbol{d}$ by adding a root of label $d$ to the $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, where $d$ is chosen according to the guesses made by $E$: $d$ is equal to the data value of the root of $\boldsymbol{d}_1$ if $E$ guessed so, to the data value of the root of $\boldsymbol{d}_2$ if $E$ guessed so, or to an arbitrary data value of $\boldsymbol{d}_1$ or $\boldsymbol{d}_2$ whose associated $B$-states are the one guessed by $E$.

Let us show that in the above detailed case, $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ has the desired properties. Notice that the presence of the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$ guarantees that data values of $x$, $x_1$ and $x_2$ are equal in $\boldsymbol{d}$ (recall that $p_1$ and $p_2$ are the root-states of $q_1$ and $q_2$). Let $\boldsymbol{a}'$ be the outcome of the run of $A$ on $\boldsymbol{a}$ as simulated by $E$, and let $\boldsymbol{a}'_1$ and $\boldsymbol{a}'_2$ be its left and right child. Let $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}$. Let $e$ be a data value occurring in $\boldsymbol{d}$.

For $e = d$, the root symbol of $\boldsymbol{t}'[e]{\downarrow_\#}$ is $a'$ and $C_p$ is increased by 1 as expected, thanks to the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$. By induction hypothesis ($\diamond$), and since $B$ is deterministic and complete, $f(p) = 1$ and for all other non #-state the corresponding value via $f$ will be 0. Hence $p$ is the new root-state of $q$ and $f$ and ($\diamond$) holds.

For $e \neq d$ we consider 3 subcases. If $e$ occurs in both $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ then $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}_1, \boldsymbol{s}_2)$ for some non empty forests $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Let $v_1$ and $v_2$ be the states reached by $B$ when evaluating $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, they are the $B$-states associated to $e$ in $\boldsymbol{a}'_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{a}'_2 \otimes \boldsymbol{d}_2$. By construction of $\boldsymbol{d}$, there are $n_\theta$ such data values $e$, where $\theta = C_{v_1} \ominus C_{v_2} \to C_v$ for the appropriate $v$. These $n_\theta$ data values will contribute to an increase of $C_v$ by $n_\theta$ as expected.

Assume now that $e$ occurs in $\boldsymbol{d}_1$ but not in $\boldsymbol{d}_2$ (the remaining case being symmetrical). Then $\boldsymbol{t}'[e]$ has the form $\#(\#(\boldsymbol{s}), \#)$ where $\boldsymbol{s}$ is some non empty forest and therefore $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}')$ (see Example 1). Hence the state associated to $e$ on $\boldsymbol{t}'$ is the same as the one associated to $e$ on $\boldsymbol{t}_1$. This is consistent with the behavior of $E$ that propagates upward the value of the counter corresponding to this state, after applying the constraints.

Altogether this shows that $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ verifies ($\star$). □

## 4 From Counter Tree Automata to $\mathbf{FO}^2(<, +1, \sim)$

We show in this section that reachability of EBVASS can be expressed as a sentence of $\mathrm{FO}^2(<, +1, \sim)$. This concludes the loop of reductions, showing that reachability for EBVASS, satisfiability of $\mathrm{FO}^2(<, +1, \sim)$ and emptiness of $\mathrm{DA}^\#$ are equivalent as decision problems.

**Theorem 3.** *The reachability problem for* EBVASS *reduces to the satisfiability problem for* $\mathrm{FO}^2(<, +1, \sim)$.

*Proof (sketch).* Given an EBVASS $E = (\mathbb{A}, Q, q_0, k, \delta)$ and a state $p \in Q$, we compute a sentence $\phi \in \mathrm{FO}^2(<, +1, \sim)$ such that $\phi$ has a model iff the configuration $(p, f_0)$ is reachable. We associate to $E$ the following finite alphabet $\mathbb{A}_E = \delta \cup \{D_i, I_i \mid 1 \leq i \leq k\} \cup \{T_\theta, L_\theta, R_\theta \mid \theta \in \Theta, (p_1, a, p_2, p, \Theta) \in \delta\}$.
Intuitively $D_i$ says that the counter $i$ has been decreased, $I_i$ says that the counter $i$ has been increased, the letters $L_\theta$, $R_\theta$ and $T_\theta$ will be used to enforce the constraint $\theta$. The formula $\phi$ will essentially accept binary trees of $\mathrm{Trees}(\mathbb{A}_E \times \mathbb{D})$ encoding runs of $E$ (we will see that $\phi$ will also accepts some other trees).
We start by the encoding of a single transition $\mu \in \delta$.

If $\mu$ is an $\epsilon$-transition then we encode it with two nodes $x, y$ where $y$ is the unique child of $x$ and the label of $x$ is $\mu$ while the label of $y$ is $D_i$ (resp. $I_i$) if $\mu$ was decreasing (resp. increasing) counter $i$.

If $\mu$ is a up-transition with set of constraints $\Theta$, then we encode it using a subtree of this form:

- The root has label $\mu$,
- below the root there is a (vertical) sequence of nodes of arity one whose labels form a word of $\Sigma_{\theta \in \Theta, \theta = C_{i_1} \ominus C_{i_2} \to C_i} (I_i T_\theta)^*$, where $\Sigma$ denotes concatenation,
- the last node of that sequence has arity two and two branches starts from that node: the left one and the right one,
- the sequence of labels of the left branch form a word of $\Sigma_{\theta \in \Theta, \theta = C_{i_1} \ominus C_{i_2} \to C_i} (D_{i_1} L_\theta)^*$,
- the sequence of labels of the right branch form a word of $\Sigma_{\theta \in \Theta, \theta = C_{i_1} \ominus C_{i_2} \to C_i} (D_{i_2} R_\theta)^*$,
- for all $\theta$ the number of occurrences of $L_\theta, R_\theta$ and $T_\theta$ are the same.

If all these items but the last one are satisfied, we say that the resulting tree is a pseudo-encoding of the up-transition $\mu$. Notice that pseudo-encodings of up-transitions form a regular tree language.

From there, the encoding of a run is obtained in the obvious way by concatenating encodings of transitions. The formula $\phi$ essentially describes this construction. It first enforces that the tree has the desired shape:

- The tree is a repetition of a sequence of the form: a pseudo-encoding of one up-transitions followed by the encodings of several $\epsilon$-transitions
- the sequencing is valid: if $\mu$ and $\nu$ are consecutive transitions in the tree then the starting state of one is the ending state of the other,
- the state $q_0$ can be found at the leaves and the state $p$ is reached at the root.

Note that the above three conditions can be checked by a standard tree automaton over $\mathbb{A}_E$, and therefore can be expressed in EMSO where the forest order part is on $\mathrm{FO}^2(<, +1, \sim)$. Therefore, by setting $\mathbb{A} = \mathbb{A}_c \times \mathbb{A}'$ for a suitable $\mathbb{A}'$ matching the existential part of the EMSO formula, the property above can be expressed in $\mathrm{FO}^2(<, +1, \sim)$.

The formula now needs to make sure that no counter ever get negative and that pseudo-encodings of up-transitions are actually real encodings. This is where data values enter the play.
The formula $\phi$ enforces that

1. no two nodes with label $D_i$ for $1 \leq i \leq k$ can have the same data value,
2. same thing with the labels $I_i$,
3. for all $i \in [k]$, every node with label $D_i$ has a descendant with label $I_i$ and with the same data value,
4. for all $i \in [k]$, every node with label $I_i$ has an ancestor with label $D_i$ and with the same data value.

These four conditions enforce that the counters never get negative and that they are all set to 0 at the root. It remains to enforce that all pseudo-encodings can be transformed into real encodings. This is done with the following conditions.

5. no two nodes with label $T_\theta$ for $\theta \in \chi$ can have the same data value,
6. same thing with the labels $L_\theta$ and $R_\theta$,
7. every node with label $T_\theta$ has a descendant with label $L_\theta$ and with the same data value,
8. every node with label $T_\theta$ has a descendant with label $R_\theta$ and with the same data value,
9. every node with label $L_\theta$ has a ancestor with label $T_\theta$ and with the same data value,
10. every node with label $R_\theta$ has a ancestor with label $T_\theta$ and with the same data value.

It is now easy to show that $\phi$ has the desired property: $\phi$ has a model iff $(p, f_0)$ is reachable by $E$, see Appendix E for the details.

## 5 Undecidability of Stronger Models

A DA$^{\#}$ $(A, B)$, as defined in Section 1, runs over data forests in two steps: the first step is a relabeling specified by the transducer $A$, the second step is the computation of $B$ on "projections" of the resulting forest, one for each data value. The projection of a data forest $\boldsymbol{t}$ on a data value $d \in \mathbb{D}$ is also defined in two steps: $(i)$ a relabeling using # for those nodes whose data value is not $d$, denoted $\boldsymbol{t}[d]$, $(ii)$ the deletion of contiguous #-nodes in $\boldsymbol{t}[d]$ as defined by some simple rewriting rules, resulting in $\boldsymbol{t}[d]{\downarrow_{\#}}$.

This is in contrast with the Data Automata defined in [5] which runs on data words but each projection does not remove consecutive #. In other terms, Step $(ii)$ is not performed. We show in this section that suppressing Step $(ii)$ would make our automata model undecidable.
More formally, a strong forest data automaton (DA$^\flat$) is defined as for DA$^{\#}$ but with the following semantic: A data forest $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d} \in \text{Forests}(\mathbb{A} \times \mathbb{D})$, is accepted by a given DA$^\flat$ $D = (A, B)$ if:

1. there exists a forest $\boldsymbol{a}' \in \text{Forests}(\mathbb{B})$ such that $\boldsymbol{a}'$ is a possible output of $A$ on $\boldsymbol{a}$ and,
2. for all $d \in \mathbb{D}$, $(\boldsymbol{a}' \otimes \boldsymbol{d})[d]$ is accepted by $B$.

The proof of the following result is given in Appendix F.

**Proposition 1.** *Emptiness of* DA$^\flat$ *is undecidable.*

## 6 Conclusion

We have seen that satisfiability of $\mathrm{FO}^2(<, +1, \sim)$, emptiness of $\mathrm{DA}^{\#}$ and reachability of EBVASS are equivalent problems in terms of decidability. The main open problem is of course whether they are all decidable or not.

The use of the EBVASS constraints of the form $C_{i_1} \ominus C_{i_2} \to C_i$ is crucial for the construction of Section 3. It seems that their semantics cannot be simulated with the usual BVASS, but it remains to be formally established wether EBVASS are strictly more expressive that BVASS, and wether this extension is needed in order to capture the expressive power of $\mathrm{FO}^2(<, +1, \sim)$ on data trees.

In [2] it was shown that, over data words, the Data Automata model of [5] is more expressive than the Register Automata of [14]. It is not obvious that our automata model $\mathrm{DA}^{\#}$ captures the expressive power of the straightforward extension of register automata to data trees. This remains to be investigated.

## References


1. R. Alur, P. Černý, and S. Weinstein. Algorithmic analysis of array-accessing programs. *ACM Trans. Comput. Logic*, 13(3):27:1–27:29, Aug. 2012.
2. H. Björklund and T. Schwentick. On notions of regularity for data languages. *Theoretical Computer Science*, 411(4-5):702–715, 2010.
3. M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. In *Symp. on Logic in Computer Science (LICS'10)*, pages 243–252, 2010.
4. M. Bojańczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *Journal of the ACM*, 56(3):13:1–13:48, May 2009.
5. M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27, 2011.
6. B. Bollig, A. Cyriac, P. Gastin, and K. Narayan Kumar. Model checking languages of data words. In *Int. Conf. on Foundations of Software Science and Computation Structures (FoSSaCS'12)*, volume 7213 of *LNCS*, pages 391–405. Springer, 2012.
7. P. Bouyer. A logical characterization of data languages. *Inf. Process. Lett.*, 84(2):75–85, Oct. 2002.
8. J.-L. Coquidé and R. Gilleron. Proofs and reachability problem for ground rewrite systems. In *Proc. IMYCS'90*, Smolenice Castle, Czechoslovakia, Nov. 1990.
9. S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), Apr. 2009.
10. D. Figueira. Satisfiability of downward XPath with data equality tests. In *Symp. on Principles of Database Systems (PODS'09)*, 2009.
11. D. Figueira. Forward-XPath and extended register automata on data-trees. In *Intl. Conf. on Database Theory (ICDT'10)*, 2010.
12. D. Figueira and L. Segoufin. Bottom-up automata on data trees and vertical XPath. In *Theoretical Aspects of Computer Science (STACS'11)*, 2011.
13. M. Jurdzinski and R. Lazic. Alternating automata on data trees and XPath satisfiability. *ACM Transactions on Computational Logic*, 12(3):19, 2011.
14. M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.
15. S. Schmitz. On the computational complexity of dominance links in grammatical formalisms. In *Annual Meeting of the Association for Computational Linguistics (ACL'10)*, pages 514–524. ACL Press, 2010.

# Appendix

## A Intermediate Normal Form

We show here that every $\mathrm{FO}^2(<, +1, \sim)$ formula can be transformed into an equivalent $\mathrm{MSO}^2(<, +1, \sim)$ formula in *intermediate normal form*:

$$\exists S_1 \cdots \exists S_k \bigwedge_i \chi_i$$

where each $\chi_i$ has one of the following forms:

$$(1)\ \forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge \delta(x,y) \rightarrow \gamma(x,y) \tag{3}$$

$$(2)\ \forall x \exists y\ \alpha(x) \rightarrow (\beta(y) \wedge \delta(x,y) \wedge \epsilon(x,y)) \tag{4}$$

where each of $\alpha$ and $\beta$ is a *type*: conjunction of monadic predicates or their negation (these unary predicates are either from $\mathbb{A}$ or from $S_1, \ldots, S_k$, *i.e.* introduced by the existentially quantified variables), $\delta(x,y)$ is either $x \sim y$ or $x \not\sim y$, $\gamma(x,y)$ is one of $\neg E_{\rightrightarrows}(x,y)$, $\neg E_{\downdownarrows}(x,y)$ or $\neg(x \parallel y)$, and $\epsilon(x,y)$ is one of $x = y$, $E_{\rightarrow}(x,y)$, $E_{\rightarrow}(y,x)$, $E_{\downarrow}(x,y)$, $E_{\downarrow}(y,x)$, $E_{\rightrightarrows}(x,y)$, $E_{\rightrightarrows}(y,x)$, $E_{\downdownarrows}(x,y)$, $E_{\downdownarrows}(y,x)$, $x \parallel y$ or *false*.

This normal form is obtained by simple syntactical manipulation and is very similar to the one given in [5] for the data words case.

*Scott normal form.* We first transform the formula into Scott Normal Form obtaining a formula of the form:

$$\psi = \exists S_1 \ldots \exists S_m\ \forall x \forall y\ \chi \wedge \bigwedge_i \forall x \exists y\ \chi_i$$

where $\chi$ and every $\chi_i$ are quantifier free, and $S_1, \ldots S_m$ are new unary predicates. This transformation is folklore: a new unary predicate $S_\theta$ is introduced for each subformula $\theta(x)$ with one free variable for marking the nodes where the subformula holds. The subformula $\theta(x)$ is then replaced by $S_\theta(x)$ and a conjunct $\forall x(S_\theta(x) \leftrightarrow \theta(x)$ is added. Eventually this yields the desired normal form.

*From Scott to intermediate normal form.* We show next that every conjunct of the core of the formula in Scott Normal Form can be replaced by an equivalent conjunction of formulas of the form (1) or (2), possibly by adding new monadic quantifications upfront.

*Case* $\forall x \forall y\ \chi$ . Recall that with our definition, the relations $E_{\rightarrow}$, $E_{\rightrightarrows}$, $E_{\downarrow}$, $E_{\downdownarrows}$, $\parallel$ and equality are pairwise disjoint. Hence we can rewrite $\forall x \forall y\ \chi$ into an equivalent $\mathrm{FO}^2(<, +1, \sim)$ formula in the following form,

$$\begin{array}{lll} \forall x \forall y & \big( & x = y \rightarrow \psi_{=}(x,y) \\ & & \wedge\ E_{\rightarrow}(x,y) \rightarrow \psi_{\rightarrow}(x,y) \\ & & \wedge\ \ E_{\downarrow}(x,y) \rightarrow \psi_{\downarrow}(x,y) \\ & & \wedge\ E_{\rightrightarrows}(x,y) \rightarrow \psi_{\rightrightarrows}(x,y) \\ & & \wedge\ E_{\downdownarrows}(x,y) \rightarrow \psi_{\downdownarrows}(x,y) \\ & & \wedge\ \ x \parallel y \rightarrow \psi_{\parallel}(x,y)\big) \end{array}$$

where every formula $\psi$ is quantifier free and only involves the predicate $\sim$ together with monadic predicates. They can be obtained from $\chi$ in the obvious way.

The resulting formula is equivalent to the conjunction

$$\begin{array}{l} \quad \forall x \exists y \ \big(x = y \wedge \psi_{=}(x,y)\big) \\ \wedge\ \forall x \exists y \ \big(\neg \mathsf{last}(x) \rightarrow (E_{\rightarrow}(x,y) \wedge \psi_{\rightarrow}(x,y))\big) \\ \wedge\ \forall x \exists y \ \big(\neg \mathsf{leaf}(x) \rightarrow (E_{\downarrow}(x,y) \wedge \psi_{\downarrow}(x,y))\big) \\ \wedge\ \forall x \forall y \ E_{\rightrightarrows}(x,y) \rightarrow \psi_{\rightrightarrows}(x,y) \\ \wedge\ \forall x \forall y \ E_{\downdownarrows}(x,y) \rightarrow \psi_{\downdownarrows}(x,y) \\ \wedge\ \forall x \forall y \ x \parallel y \rightarrow \psi_{\parallel}(x,y) \end{array}$$

where $\mathsf{leaf}(x)$ stands for $\neg \exists y \ E_{\downarrow}(x,y)$, and $\mathsf{last}(x)$ for $(\exists y \ E_{\downarrow}(y,x)) \wedge (\neg \exists y \ E_{\rightarrow}(x,y))$. The 3 first conjuncts, with quantifier prefix $\forall x \exists y$, will be treated later when dealing with the second case.

For the next 3 conjuncts, putting $\neg\psi_{\rightrightarrows}$, $\neg\psi_{\downdownarrows}$, $\neg\psi_{\parallel}$ in disjunctive normal form (with an exponential blowup), we rewrite $\psi_{\rightrightarrows}$, $\psi_{\downdownarrows}$, $\psi_{\parallel}$ as a conjunction of formulas of the form $\neg(\alpha(x) \wedge \beta(y) \wedge \delta(x,y))$, where $\delta(x,y)$ is $x \sim y$ or $x \not\sim y$. By distribution of conjunction over implication, and by contraposition, we obtain for the 3 cases an equivalent conjunction of formulas of the following form (matching the desired form (1))

$$\begin{array}{l} \forall x \forall y \ \alpha(x) \wedge \beta(y) \wedge \delta(x,y) \rightarrow \neg E_{\rightrightarrows}(x,y) \\ \forall x \forall y \ \alpha(x) \wedge \beta(y) \wedge \delta(x,y) \rightarrow \neg E_{\downdownarrows}(x,y) \\ \forall x \forall y \ \alpha(x) \wedge \beta(y) \wedge \delta(x,y) \rightarrow \neg (x \parallel y) \end{array}$$

*Case* $\forall x \exists y \ \chi$ . We first transform (with an exponential blowup) $\chi$ into an equivalent disjunction of the form

$$\chi' = \bigvee_j \alpha_j(x) \wedge \beta_j(y) \wedge \delta_j(x,y) \wedge \epsilon_j(x,y)$$

where $\alpha_j$, $\beta_j$, $\delta_j$ and $\epsilon_j$ are as in (2). Next, in order to eliminate the disjunctions, we add a new monadic second-order variables $S_{\chi,j}$, that we existentially quantify upfront of the global formula, and transform $\forall x \exists y \ \chi'$ into the conjunction

$$\bigwedge_j \forall x \exists y \ (\alpha_j(x) \wedge S_{\chi,j}(x)) \rightarrow (\beta_j(y) \wedge \delta_j(x,y) \wedge \epsilon_j(x,y)) \wedge \forall x \exists y \ \big(\bigvee_j S_{\chi,j}(x)\big)$$

The first conjuncts express that if $S_{\chi,j}(x)$ holds, then there exists a node $y$ such that the corresponding conjunct of $\chi'$ holds, and the last conjunct expresses that for all node $x$, at least one of the $S_{\chi,j}(x)$ must hold and can be rewritten as $\forall x \exists y \ \big(\bigwedge \neg S_{\chi,j}(x) \rightarrow \mathit{false}\big)$. Now all the conjuncts are as in (2) and we are done.

## B Case analysis for constructing a DA$^{\#}$ from a simple FO$^2(<, +1, \sim)$ formula

We provide here the complete case analysis for transforming simple formulas of FO$^2(<, +1, \sim)$ of the form (1) or (2) into DA$^{\#}$.

Given a data tree, a *vertical path* is a set of nodes containing exactly one leaf and all its ancestors and nothing else. A *horizontal path* is a set of nodes containing one node together with all its siblings and nothing else.

We start with formulas of the form (1).

*Case 1:* $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \sim y \rightarrow \gamma(x,y)$, where $\gamma(x,y)$ is as in (1). These formulas expresses a property of pairs of nodes with the same data value and this property can be tested by the forest automaton $B$. Indeed, for each data value $d$, the relations $E_{\Rightarrow}$, $E_{\Downarrow}$ and $\|$ are preserved for pairs of nodes sharing a data value when going from $\boldsymbol{t}$ to $\boldsymbol{t}[d]{\downarrow_{\#}}$. Therefore, it is a first-order, hence regular, property over $\boldsymbol{t}[d]{\downarrow_{\#}}$.

*Case 2:* $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \neg E_{\Rightarrow}(x,y)$. This formula expresses that a data tree cannot contain an $\alpha$-node having a $\beta$-node with a different data value as a sibling to its right, except if it is the next-sibling. Let $X$ be an horizontal path in a data tree $\boldsymbol{t}$ containing at least one $\alpha$-node, and let $x$ be the leftmost $\alpha$-node in $X$. Let $d$ be the data value of $x$. Consider now a $\beta$-node $y$ of $X$ whose data value is not $d$. Clearly $y$ cannot be such that $E_{\Rightarrow}(x,y)$ otherwise the formula would be false. For any other $\beta$-node $y$ of $X$ whose data value is $d$, the pair $(x,y)$ cannot make the formula false. Moreover, every $\alpha$-node $x'$ in $X$ other than $x$ and with a data value different from $d$ cannot be such that $E_{\Rightarrow}(x',y)$.
With this discussion in mind we construct $(A,B)$ as follows. In every horizontal path $X$ containing one $\alpha$-node, the transducer $A$ identify the leftmost occurrence $x$ of an $\alpha$-node in $X$, and marks its with a new color $c'$, and marks all the nodes of $X$ with the same data as $x$ with a color $c$. As in Remark 1, the forest automaton $B$ checks that the guesses are correct, *i.e.* it accepts only forests in which every horizontal path $X$ satisfy one of the following conditions: $X$ contains one occurrence of the color $c'$ and all other nodes of $X$ not labelled with # are marked with $c$, or $X$ contains none of the new colors at all.

The transducer $A$ rejects if there are some unmarked $\beta$-nodes occurring as a right sibling (except for the next-sibling) of a node marked with $c'$. Moreover, $A$ checks that the formula is correct for all marked $\beta$-nodes, *i.e.* there is no unmarked $\alpha$-node as left sibling, except for the previous sibling, of a marked $\beta$-node. As explained in Remark 1, this is a regular property.

*Case 3:* $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \neg E_{\Downarrow}(x,y)$. The property expressed by this formula is similar to the previous case, replacing the right sibling relationship with the descendant relationship.

Let $X$ be a vertical path in a data tree $\boldsymbol{t}$ containing at least one $\alpha$-node, and let $x$ be the $\alpha$-node in $X$ the closest to the root. Let $d$ be the data value of $x$. Consider now a $\beta$-node $y$ of $X$ whose data value is not $d$. Clearly $y$ cannot be such that $E_{\Downarrow}(x,y)$ otherwise the formula would be false. For any other $\beta$-node $y$ of $\boldsymbol{t}$, the pair $(x,y)$ cannot make the formula false. Moreover, every $\alpha$-node $x'$ in $X$ other than $x$ and with a data value different from $d$ cannot be such that $E_{\Downarrow}(x',y)$.

The construction of $(A, B)$ is similar to the previous case, except that different vertical paths may share some nodes. The transducer $A$ marks all the $\alpha$-nodes that have no $\alpha$-node as ancestor, with a new color $c'$. Then, for every node $x$ marked $c'$, $A$ guesses all the nodes inside the subtree rooted at $x$ having the same data value as $x$ and mark them with a new color $c$. As in Remark 1, the forest automaton $B$ checks that the guesses of colors are correct for each vertical path, as in the previous case.

The transducer $A$ rejects if there is some unmarked $\beta$-nodes that is a descendant but not a child of a node of color $c'$. Moreover it tests the formula for all marked $\beta$-nodes, checking that there is no unmarked $\alpha$-node as ancestor, except for the parent node. This is a again a regular property, as explained in Remark 1.

*Case 4:* $\forall x \forall y\ \alpha(x) \wedge \beta(y) \wedge x \not\sim y \rightarrow \neg(x \parallel y)$. The property expresses that every two nodes of type respectively $\alpha$ and $\beta$ and with different data values cannot be parallel. Recall that two nodes are parallel if they are not parents and not siblings.

*Subcase 4.1:* There exists two $\alpha$-nodes that are parallel.

Let $x_1$ and $x_2$ be two parallel $\alpha$-nodes and let $z$ be their least common ancestor. We can choose $x_1$ and $x_2$ such that none of the $\alpha$-nodes are parallel to $z$ or sibling of $z$, because if this was not the case then there is an $\alpha$-node $x_3$ parallel to $z$ or sibling of $z$, and therefore $x_3$ is parallel to $x_1$, and we can replace $x_2$ with $x_3$, continuing with a strictly higher node $z$. Let $z_1$ and $z_2$ be the children of $z$ that are respectively ancestors of $x_1$ and $x_2$. Note that by construction, $z_1 \neq z_2$. If $x_1 = z_1$ and there is an $\alpha$-node $x_3$ in the subtree of $z$, different from $x_1$ and parallel to $z_2$, then we replace $x_1$ by $x_3$ and proceed. In other words we ensure that if $x_1 = z_1$ then there is no $\alpha$-node parallel to $z_2$ in the subtree of $z$. We do the same trick to enforce that if $x_2 = z_2$ then there is no $\alpha$-node parallel to $z_1$ in the subtree of $z$. Notice that we cannot have at the same time $x_1 = z_1$ and $x_2 = z_2$. All these properties can be specified in MSO and therefore can be tested by a tree automaton. Let $d_1$ and $d_2$ be the respective data values of $x_1$ and $x_2$ (possibly $d_1 = d_2$).

Consider now a $\beta$-node $y$ whose data value is neither $d_1$ nor $d_2$. If $y$ is parallel to $z$ or sibling of $z$, then the formula cannot be true as it is contradicted by $(x_1, y)$. If $y$ is an ancestor of $z$ then, as no $\alpha$-node is parallel to $z$, none is parallel to $y$, and hence the formula cannot be falsified with this $y$. Assume now that $y$ is inside the subtree of $z$. If $y = z_1$ and $x_2 \neq z_2$, then the formula is contradicted by $(x_2, y)$. If $y = z_1$ and $x_2 = z_2$, then, by hypothesis, there is no $\alpha$-node parallel to $y$ in the subtree of $z$, and there is no $\alpha$-node parallel to $y$ outside the subtree of $z$, and altogether, and the formula holds for $y$. If $y \neq z_1$ and $y$ is a descendant of $z_1$, then the formula is contradicted by $(x_2, y)$. The cases where $y$ is descendant of $y_2$ are symmetric: in this case, the formula can only be true if $y = y_2$ and $x_1 = z_1$. In the remaining cases $y$ is in the subtree of $z$ and not in the subtrees of $y_1$ and $y_2$ make the formula false. Indeed, in each of these cases, either $(x_1, y)$ or $(x_2, y)$ contradicts the formula.

With this discussion in mind, this case can be solved as follows: The transducer $A$ guesses the nodes of $x_1$, $x_2$, $z_1$, $z_2$ and $z$ and checks that they satisfy the appropriate properties. Moreover, $A$ guesses whether $d_1 = d_2$ and marks accordingly the data values of $x_1$ and $x_2$ with two or one new colors. The forest automaton $B$ will then check that the data values are marked appropriately as in Remark 1.

Moreover $A$ checks that for all marked $\beta$-nodes there is no $\alpha$-node parallel to it and with a different data value, a regular property as explained in Remark 1. It now remains for $A$ to check that every unmarked $\beta$-node $y$ behave according to the discussion above: $y$ is an ancestor of $z$ or $y = z_1$ and $x_2 = z_2$ or $y = z_2$ and $x_1 = z_1$. This is a regular property testable by $A$.

*Subcase 4.2:* There are no two parallel $\alpha$-nodes.

Let $x$ be an $\alpha$-node such that no $\alpha$-node is a descendant of $x$. By hypothesis, all $\alpha$-nodes are either ancestors or siblings of $x$. Let $d$ be the data value of $x$. We distinguish between several subcases depending on whether there are other $\alpha$-nodes that are siblings of $x$ or not.

If there is an $\alpha$-node $x'$ that is a sibling of $x$, then let let $d'$ be its data value (possibly $d = d'$). Note that by hypothesis in this case, every $\alpha$-node is either a sibling of $x$ or an ancestor $x$. Consider now a $\beta$-node $y$ whose data value is neither $d$ nor $d'$. Then, in order to make the formal true, $y$ must be an ancestor or a sibling of $x$.

In this case, the transducer $A$ guesses the nodes $x$ and $x'$ and marks the corresponding data values with one or two new colors (according to whether $d = d'$ or not). The forest automaton $B$ will then check that the data values are marked correctly as explained in Remark 1. For the marked $\beta$-nodes, the property is regular and can also be checked by $A$. It remains for $A$ to check that every unmarked $\beta$-nodes is either an ancestor of $x$ or a sibling of $x$.

Now, if there are no $\alpha$-nodes that are sibling of $x$, and $y$ is a $\beta$-node whose data value is not $d$, then in order to make the formal true, $y$ cannot be parallel to $x$, and therefore, $y$ can be an ancestor, a descendant or a sibling of $x$.

In this second case, the transducer $A$ guesses the node $x$, marks its data value using a new color. The forest automaton $B$ will then check that the data values were marked correctly as explained in Remark 1. The transducer $A$ checks that all marked $\beta$-nodes make the formula true, and that all unmarked $\beta$-nodes are not parallel to $x$.

*Case 5:* $\forall x \exists y\ \alpha(x) \to (\beta(y) \wedge x \sim y \wedge \epsilon(x, y))$, where $\epsilon(x, y)$ is as in (2). These formulas express properties of nodes with the same data and therefore can be treated by the forest automaton $B$ as for the case 1.

*Case 6:* $\forall x \exists y\ \alpha(x) \to (\beta(y) \wedge x \not\sim y \wedge E_{\to}(x, y))$. The transducer $A$ marks every $\alpha$-node $x$, with a new color $c$ and checks that the next-sibling $x$ is a $\beta$-node. The forest automaton $B$ accepts only the forests such that for every node marked with $c$, its right sibling is labeled with $\#$. The cases where $\epsilon(x, y)$ is one of $E_{\to}(y, x)$, $E_{\downarrow}(x, y)$, $E_{\downarrow}(y, x)$ are treated similarly.

*Case 7:* $\forall x \exists y\ \alpha(x) \to (\beta(y) \wedge x \not\sim y \wedge E_{\rightrightarrows}(x, y))$. This property expresses that every $\alpha$-node must have a $\beta$-node as a right sibling, but not as its next-sibling, and with a different data value.

Let $X$ be an horizontal path. Let $y$ be the rightmost $\beta$-node of $X$ and $d$ be its data value. Consider now an $\alpha$-node $x$ of $X$ with a data value different from $d$. Then either $x$ is to the left of the previous-sibling of $x$ and $y$ can serve as the desired witness, or $x$ has no witness and the formula is false.

The transducer automaton $A$, for each horizontal path $X$ containing an $\alpha$-node, marks its rightmost $\beta$-node $y$ with a new color $c'$, guesses all the nodes of $X$ with the same data value as $y$ and marks them with a new color $c$. Then it checks that all unmarked $\alpha$-node of $X$ occurs to the left of the previous-sibling of $y$. For the all marked $\alpha$-nodes, it also tests the appropriate regular property. The forest automaton $B$ checks that for each horizontal paths, either all elements are marked with $c$ or $c'$, or none.

*Cases 8, 9, 10:* The constructions for the cases where $\epsilon(x, y)$ is one of $E_{\rightrightarrows}(y, x))$, $E_{\downdownarrows}(x, y))$, and $E_{\downdownarrows}(y, x))$ are similar.

*Case 11:* $\forall x \exists y\ \alpha(x) \to (\beta(y) \wedge x \not\sim y \wedge x \parallel y)$. This property expresses that every $\alpha$-node must have a parallel $\beta$-node with a different data value.

*Subcase 11.1:* There exists two $\beta$-nodes that are parallel.

Let $y_1$ and $y_2$ be two parallel $\beta$-nodes and let $z$ be their least common ancestor. Using the same reasoning as in subcase 4.1, we can choose $y_1$ and $y_2$ such that none of the $\beta$-nodes is parallel to $z$ or a sibling of $z$. Let $z_1$ and $z_2$ be the children of $z$ that are the ancestors of $y_1$ and $y_2$ respectively. By construction, $z_1 \neq z_2$. Using the same trick as in subcase 4.1, we can ensure that if $y_1 = z_1$ then there is no $\beta$-node parallel to $z_2$, and if $y_2 = z_2$ then there is no $\beta$-node parallel to $z_1$. Moreover, we cannot have at the same time $y_1 = z_1$ and $y_2 = z_2$. Recall that all these properties can be tested by a tree automaton. Let $d_1$ and $d_2$ be the respective data values of $y_1$ and $y_2$ (possibly $d_1 = d_2$).

Consider now an $\alpha$-node $x$ whose data value is neither $d_1$ nor $d_2$. If $x$ is parallel to $z$ or a sibling of $z$, then $y_1$ is a witness for $x$. If $x$ is an ancestor of $z$ then by hypothesis there is no $\beta$-node parallel to $x$ and hence the formula is false. Assume now that $x$ is in the subtree rooted at $z$. If $x = z_1$ and $y_2 \neq z_2$, then $y_2$ is a $\beta$-node parallel to $x$ with a different data value, hence a witness for $x$ in the formula. If $x = z_1$ and $y_2 = z_2$, then by hypothesis, there is no $\beta$-node parallel to $x$ in the subtree of $z$, and since there are neither $\beta$-nodes parallel to $x$ outside the subtree of $z$, the formula must be false. If $x \neq z_1$ and $x$ is a descendant of $z_1$, then $y_2$ is a witness for $x$. The cases where $x$ is a descendant of $z_2$ are symmetric. In the remaining cases, $x$ is in the subtree of $z$ and not a descendant of $z_1$ or $z_2$. In each of these cases, either $y_1$ or $y_2$ is a witness for $x$.

With this discussion in mind, this case can be solved as follows: The transducer $A$ guesses the nodes of $y_1$, $y_2$, $z_1$, $z_2$ and $z$ and check that they satisfy the appropriate properties. Moreover, $A$ guesses whether $d_1 = d_2$ and marks accordingly the data values of $z_1$ and $z_2$ with one or two new color. The forest

automaton $B$ will then check that the data values are marked appropriately, as in Remark 1. Moreover $A$ checks that for every marked $\alpha$-node, there exists a $\beta$-node making the formula true. It remains for $A$ to check that no unmarked $\alpha$-node occurs above $z$, that if $y_1 = z_1$ then $z_2$ is not an unmarked $\alpha$-node and that if $y_2 = z_2$ then $z_1$ is not an unmarked $\alpha$-node.

*Subcase 11.2:* There are no two $\beta$-nodes that are parallel.

Let $y$ be an $\beta$-node such that no $\beta$-node is a descendant of $y$. By hypothesis, all $\beta$-nodes are either ancestors or siblings of $y$. Let $d$ be the data value of $y$. We distinguish between several subcases depending on whether there are $\beta$-nodes that are siblings of $y$ or not.

If there exists a $\beta$-node $y'$ that is a sibling of $y$, let $d'$ be its data value (possibly $d = d'$). Consider an $\alpha$-node $x$ whose data value is neither $d$ nor $d'$. If $x$ is parallel to $y$, then we have have a witness $(y)$ for $x$. If $x$ is an ancestor or a sibling of $y$, then the formula cannot be true, because by hypothesis every $\beta$-node cannot be parallel to $x$. If $x$ is a descendant of $y$, then $y'$ makes the formula true for that $x$.

Consider now the case where there are no $\beta$-node that are sibling of $y$. Note that $y$ can have $\beta$-nodes amongst its ancestors. Let $x$ be a $\alpha$-node that has data value different from $d$. If $x$ is not parallel to $y$ then the formula must be false. Otherwise, $y$ is a witness for $x$.

The transducer $A$ guesses the $\beta$-node $y$ and marks its data value using a new color. Then it checks whether there is an $\beta$-node $y'$ that is a sibling of $y$. If yes, it guesses whether the value at $y'$ is the same as the value at $y$ or not, and marks the data value of $y'$ using a new color. The forest automaton $B$ will then check that the data values are marked appropriately. For marked $\alpha$-nodes, $A$ checks the regular property making the formula true. It now remains for $A$ to check, in both cases, that every unmarked $\alpha$-node $x$ satisfy the appropriate condition described above, *i.e.* that $x$ is parallel to $y$ or a descendant of $y$ if there exists a sibling $y'$ and that $x$ is parallel to $y$ otherwise.

*Case 12:* $\forall x \exists y\ \alpha(x) \rightarrow \mathit{false}$. It is sufficient to test with $A$ that no $\alpha$-node is present in the tree.

## C Reduction to the binary data tree case (Lemma 2)

It is well known that unranked trees can be transformed into binary tree using the first-child/next-sibling encoding and that this point of view preserves regularity. We use a simple extension to data trees of this encoding. This mapping called $\mathsf{fcnsd}$ transforms a forest of $\mathrm{Forests}(\mathbb{A}_{\#} \times \mathbb{D})$ into a binary tree of $\mathrm{Trees}\big(\mathbb{A}_{\#} \times \mathbb{D}\big)$ (note that the symbol # is the same as the one used in the definition of $\boldsymbol{t}[d]{\downarrow_{\#}}$ in Section 1), as defined recursively in the following

$$
\begin{array}{lll}
\mathsf{fcnsd}\big(\langle a,d\rangle\big) & = \langle a,d\rangle\big(\langle \#,d\rangle,\langle \#,d\rangle\big) & \text{if } a \in \mathbb{A}_{\#} \\
\mathsf{fcnsd}\big(\langle a,d\rangle(\boldsymbol{t})\big) & = \langle a,d\rangle\big(\mathsf{fcnsd}(\boldsymbol{t}) + \langle \#,d\rangle\big) & \text{if } |\boldsymbol{t}| \geq 1 \\
\mathsf{fcnsd}\big(\langle a,d\rangle + \boldsymbol{t}\big) & = \langle a,d\rangle\big(\langle \#,d\rangle + \mathsf{fcnsd}(\boldsymbol{t})\big) & \text{if } |\boldsymbol{t}| \geq 1 \\
\mathsf{fcnsd}\big(\langle a,d\rangle(\boldsymbol{t}) + \boldsymbol{t}'\big) & = \langle a,d\rangle\big(\mathsf{fcnsd}(\boldsymbol{t}) + \mathsf{fcnsd}(\boldsymbol{t}')\big) & \text{if } |\boldsymbol{t}|, |\boldsymbol{t}'| \geq 1
\end{array}
$$

Forgetting the data values in the above definition of $\mathsf{fcnsd}$, we obtain the classical first-child/next-sibling mapping $\mathsf{fcns}$, which transforms a forest of Forests$(\mathbb{A}_{\#})$ into a binary tree of Trees$(\mathbb{A}_{\#})$.

**Lemma 2** Given a DA$^{\#}$ $D$, there exists a DA$^{\#}$ $D'$ accepting exactly the binary encoding of the data trees accepted by $D$. Moreover $D'$ can be effectively computed from $D$.

*Proof.* We start with some useful facts.

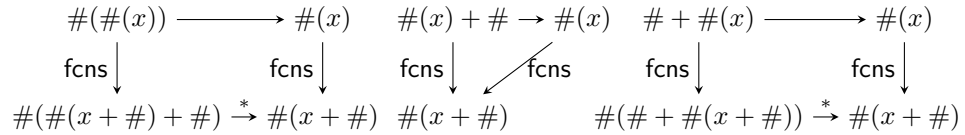
**Fact 1** *For all* $\boldsymbol{t} \in \mathit{Forests}(\mathbb{A} \times \mathbb{D})$ *and* $d \in \mathbb{D}$, $\mathsf{fcns}(\boldsymbol{t}[d]) = \big(\mathsf{fcnsd}(\boldsymbol{t})\big)[d]$.

*Proof.* By induction on the forest $\boldsymbol{t}$. If $\boldsymbol{t}$ has the form $\langle a,e\rangle$, then $\mathsf{fcnsd}(\boldsymbol{t}) = \langle a,e\rangle\big(\langle \#,e\rangle + \langle \#,e\rangle\big)$. Given $d \in \mathbb{D}$, if $d = e$, then $\boldsymbol{t}[d] = a$ and $\mathsf{fcns}(\boldsymbol{t}[d]) = a(\#,\#) = \big(\mathsf{fcnsd}(\boldsymbol{t})\big)[d]$, if $d \neq e$, $\boldsymbol{t}[d] = \#$ and $\mathsf{fcns}(\boldsymbol{t}[d]) = \#(\#,\#) = \big(\mathsf{fcnsd}(\boldsymbol{t})\big)[d]$.

If $\boldsymbol{t} = \langle a,e\rangle(\boldsymbol{t}_1)$, then $\mathsf{fcnsd}(\boldsymbol{t}) = \langle a,e\rangle\big(\mathsf{fcnsd}(\boldsymbol{t}_1) + \langle \#,e\rangle\big)$. Given $d \in \mathbb{D}$, $\boldsymbol{t}[d] = a(\boldsymbol{t}_1[d])$ if $d = e$ and $\boldsymbol{t}[d] = \#(\boldsymbol{t}_1[d])$ otherwise. In the first case $d = e$, $\mathsf{fcns}(\boldsymbol{t}[d]) = a(\mathsf{fcns}\big(\boldsymbol{t}_1[d]\big),\#) = \big(\mathsf{fcnsd}(\boldsymbol{t})\big)[d]$, because by induction hypothesis $\mathsf{fcns}(\boldsymbol{t}_1[d]) = \mathsf{fcnsd}(\boldsymbol{t}_1)[d]$, and similarly, if $d \neq e$, $\mathsf{fcns}(\boldsymbol{t}[d]) = \#(\mathsf{fcns}\big(\boldsymbol{t}_1[d]\big),\#) = \big(\mathsf{fcnsd}(\boldsymbol{t})\big)[d]$. The cases $\boldsymbol{t} = \langle a,d\rangle + \boldsymbol{t}_1$ and $\boldsymbol{t} = \langle a,d\rangle(\boldsymbol{t}_1) + \boldsymbol{t}'_1$ are similar. □

**Fact 2** *For all* $\boldsymbol{t} \in \mathit{Trees}(\mathbb{A}_{\#})$, $\big(\mathsf{fcns}(\boldsymbol{t})\big)\!\downarrow_{\#} = \mathsf{fcns}(\boldsymbol{t}\!\downarrow_{\#})\!\downarrow_{\#}$.

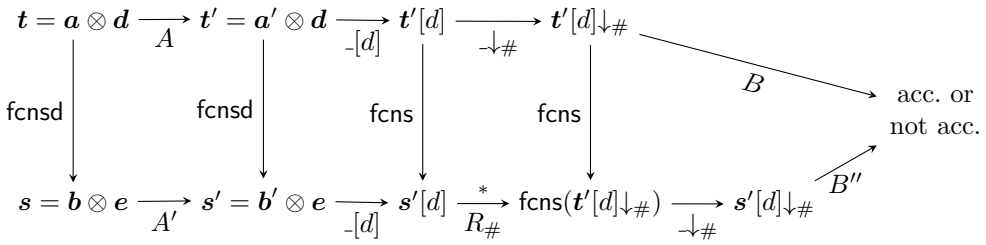
*Proof.* Let $R_{\#}$ be the rewrite system containing the three rewrite rules presented in Section 1 for the definition of $\downarrow_{\#}$, and let $\xrightarrow[R_{\#}]{}$ denote the relation of application of one rule of $R_{\#}$, and $\xrightarrow[R_{\#}]{*}$ its reflexive and transitive closure. We can observe that for all $\boldsymbol{t}, \boldsymbol{s} \in \mathrm{Trees}(\mathbb{A}_{\#})$, if $\boldsymbol{t} \xrightarrow[R_{\#}]{} \boldsymbol{s}$, then $\mathsf{fcns}(\boldsymbol{t}) \xrightarrow[R_{\#}]{*} \mathsf{fcns}(\boldsymbol{s})$.
It is sufficient to see that this property holds for each of the three rewrite rules of $R_{\#}$.




It follows that for all $\boldsymbol{t} \in \mathrm{Trees}(\mathbb{A}_{\#})$, $\mathsf{fcns}(t) \xrightarrow[R_{\#}]{*} \mathsf{fcns}(t\!\downarrow_{\#})$, and since the rewrite system $R_{\#}$ is confluent and terminating, $\mathsf{fcns}(t)\!\downarrow_{\#} = \mathsf{fcns}(t\!\downarrow_{\#})\!\downarrow_{\#}$. □

We are now ready to prove Lemma 2. Assume that $D = (A,B)$. Given a tree automata $B$, there is a classical construction of a tree automata $B'$ recognizing

the first-child/next-sibling representation of the forests accepted by $B$. It readily extends to letter-to-letter transducers. Hence from $A$ we get a transducer $A'$ working on binary trees such that from any tree $\boldsymbol{a}$ the relabeling performed by $A'$ on $\mathsf{fcns}(\boldsymbol{a})$ is the same as the first-child/next-sibling representation of the relabeling performed by $A$ on $\boldsymbol{a}$. Together with Facts 1 and 2, this makes the following diagram commute.




where the tree automaton $B''$ is constructed as follows from $B'$ (the tree automaton recognizing the first-child/next-sibling representation of the forests accepted by $B$). First, we assume that $B$ recognizes only forests in normal form wrt $R_\#$ ($R_\#$ is defined in the proof of Fact 2). This is not restrictive as the set of trees in normal forms wrt $R_\#$ is regular, hence its intersection with $B$ is also regular. Second, we compute a tree automaton $B^*$ recognizing the set of descendants under rewriting with the rules of $R_\#$ of the tree accepted by $B'$, using e.g. a construction from [8]. Finally, $B''$ is the tree automaton recognizing the intersection of the set of trees accepted by $B^*$ and the trees in normal form wrt $R_\#$. The DA$^\#$ $(A', B'')$ accepts exactly the first child/next sibling encodings of the data trees accepted by $D$. □

## D Reduction from DA$^\#$ to EBVASS (Theorem 2)

**Theorem 2** Given a DA$^\#$ $D$, there exists a EBVASS $E$ accepting exactly the trees $\boldsymbol{a}$ such that $\boldsymbol{a}\otimes\boldsymbol{d}$ is accepted by $D$ for some $\boldsymbol{d}$. Moreover $E$ can be effectively computed from $D$.

*Proof.* Assume $D = (A, B)$ is a data automata running over Trees($\mathbb{A} \times \mathbb{D}$), with $A = (Q_A, \mathbb{A}, \mathbb{B}, F_A, \Delta_A)$ and $B = (Q_B, \mathbb{B}, F_B, \Delta_B)$. To simplify the notation, for any tree $\boldsymbol{a} \in$ Trees($\mathbb{A}$) we shall write $\boldsymbol{a}' \in A(\boldsymbol{a})$ if the tree $\boldsymbol{a}' \in$ Trees($\mathbb{B}$) is a possible output of $A$ on $\boldsymbol{a}$. The forest automaton $B$ is assumed deterministic and complete, *i.e.* for every $\boldsymbol{a}' \in$ Trees($\mathbb{B} \cup \{\#\}$), $B$ evaluates into exactly one state of $Q_B$. In particular, let $p_\# \in Q_B$ be the state on which $B$ evaluates on the tree with a single node labeled with #. Moreover we also assume *wlog* that the states of $B$ indicate whether the last letter read was a # or not. In the former case we say that the state is a #-state.

To any data tree $\boldsymbol{t} \in$ Trees($\mathbb{B} \times \mathbb{D}$), and any data value $d$ occurring in $\boldsymbol{t}$, the state corresponding to the evaluation of $B$ on $\boldsymbol{t}[d]\downarrow_\#$ is called *the $B$-state associated to $d$ in $\boldsymbol{t}$*. To any data tree $\boldsymbol{t} \in$ Trees($\mathbb{B}\times\mathbb{D}$), we associate a function $K_{\boldsymbol{t}}$

from $Q_B \backslash \{p_\#\}$ into $\mathbb{N}$ such that for all $p \in Q_B \backslash \{p_\#\}$, $K_{\boldsymbol{t}}(p)$ is the number of data values having $p$ as associated $B$-state on $\boldsymbol{t}$. Note that because $B$ is deterministic and complete, there is at most one non $\#$-state $p$ such that $K_{\boldsymbol{t}}(p) \neq 0$ and actually $K_{\boldsymbol{t}}(p) = 1$. This state is the one associated to the data value of the root of $\boldsymbol{t}$. We call this special state the *root-state* of $\boldsymbol{t}$.

We now construct an EBVASS is $E = (\mathbb{A}, Q, q_0, k, \delta)$ with $k = |Q_B| - 1$ recognizing the projection on $\mathbb{A}$ of the data trees accepted by $D$. Each configuration of $E$ is then of the form $(q, f)$ where $q \in Q$ and $f$ is a function associating to each counter its value. In the remaining part of this proof we view each such function $f$ as a function from $Q_B$ to the natural numbers.

The EBVASS $E$ is constructed such that every state of $Q$ is mode of a state of $Q_A$, a state of $Q_B$, together with auxiliary information. For each $q \in Q$ we call the state of $Q_A$ contained in $q$ the $A$-state of $q$. $E$ will also ensure the following property:

($\diamond$) if $E$ can reach the configuration $(q, f)$ at a non leaf mode, then $f$ has only one non $\#$-state $p$ verifying $f(p) \neq 0$ and actually $f(p) = 1$.

We will refer to this state $p$ as the *root-state* of $f$. $E$ is constructed such that any state $q \in Q$ contains the root-state of the current configuration, denoted as the root-state of $q$ in the sequel. Finally $E$ will verify the following property which clearly imply the result:

($\star$) $E$ reaches the configuration $(q, f)$ at the root of a tree $\boldsymbol{a}$ iff there is a data tree $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ and $\boldsymbol{a}' \in A(\boldsymbol{a})$ such that $K_{\boldsymbol{a}' \otimes \boldsymbol{d}} = f|_{Q_B \backslash \{p_\#\}}$ and there is a run of $A$ witnessing $\boldsymbol{a}' \in A(\boldsymbol{a})$ whose value at the root of $\boldsymbol{a}$ is the $A$-state of $q$.

Notice that the property ($\star$) is invariant under permutations of $\mathbb{D}$. Hence if a tree $\boldsymbol{d}$ witnesses the property ($\star$), then any tree $\boldsymbol{d}'$ constructed from $\boldsymbol{d}$ by permuting the data values is also a witness for ($\star$). This observation will be useful for showing the correctness of the construction of $E$.

Intuitively $E$ works as follows. From configurations $(q_1, f_1)$ and $(q_2, f_2)$ reached respectively at the left and right children $x_1$ and $x_2$ of a node $x$ whose label is $a \in \mathbb{A}$, it does the following: it simulates $A$ using $a$ and the $A$-states of $q_1$ and $q_2$ obtaining the new $A$-state for $x$ together with the relabeling $a' \in \mathbb{B}$ of $x$. Now $E$ needs to simulate $B$ on all data values, updating the counters appropriately. In order to do this, $E$ has to "guess" the data value $d$ of $\boldsymbol{d}$ at $x$, assuming that it has already guessed the subtrees $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ of $\boldsymbol{d}$ at $x_1$ and $x_2$. Of course, $E$ cannot make a guess among an infinite set of choices, but it is enough for $E$ to guess whether the data value is a new data value or not, *i.e.* whether $d$ is present in $\boldsymbol{d}_1$ or $\boldsymbol{d}_2$. In the case $d$ is already present, $E$ guesses the $B$-state associated to $d$ in the subtrees $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ of $\boldsymbol{t}$ rooted at $x_1$ and $x_2$. Note that because $B$ is deterministic and complete, these guesses implies whether $d$ is the data value of $x_1$ or or/and of $x_2$ in $\boldsymbol{d}$. Moreover $E$ guesses whether the data value at $x_1$ occurs in the subtree $d_2$ of $x_2$ or not, and vice-versa for the data value at $x_2$. These tests make finitely many guesses and we will see that this is enough for $E$ to recover the computations on trees $(\boldsymbol{a}' \otimes \boldsymbol{d})[e]{\downarrow_\#}$ for all data values $e$.

We now work out the details. We set $Q = Q_A \times Q_B \times Q_0$, where $Q_0$ is a finite set of auxiliary control states. For any state $q \in Q$, its first component gives the $A$-state of $q$ while the second provides its root-state.
Let us now define the transitions of $E$. The simulation of $A$ is straightforward: We ensure that for every $\epsilon$-transition $(q, a, q', u)$ of $E$, the $A$-states of $q$ and $q'$ coincide, and that for every up-transition $(q_1, a, q_2, q, \Theta)$ of $E$, there exists a transition of $A$ of the form $(p_1, a, p_2, p, a')$, for some $a' \in \mathbb{B}$ such that $p_1, p_2$ and $p$ are the $A$-states of $q_1$, $q_2$ and $q$.

We now define how the counters are modified during the transitions in order to maintain the properties $(\diamond)$ and $(\star)$. For each of the guesses that $E$ made about how the data values at $x$, $x_1$ and $x_2$ are related, we define a set of constraints $\Theta$ and add the up-transitions $(q_1, a, q_2, q, \Theta)$ into $E$, where $q_1, a, q_2$ and $q$ are related as above in order to simulate $A$. We let $a'$ be the letter guessed by $A$ for relabeling the current node $x$ and let $p_1$, respectively $p_2$ be the root-state of $q_1$ and $q_2$.

Before embarking to the case analysis, a few words about correctness.
Correctness is shown by induction on the depth of the tree based on the inductive hypotheses $(\diamond)$ and $(\star)$. Immediately after defining a set $\Theta$ and adding a transition of the form $(q_1, a, q_2, q, \Theta)$ into the set of transitions of $E$, we show that using this transition maintains $(\diamond)$ and $(\star)$. We assume that $E$ reached the configuration $(q, f)$ at the root $x$ of a tree $\boldsymbol{a}$.

If $x$ is a leaf node, then by definition of EBVASS, $q$ is the initial state of $E$ and $f = f_0$ (the function setting all counters to 0), hence $(\star)$ holds.

If $x$ is an inner node, let $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ be the subtrees of $\boldsymbol{a}$ rooted at the children $x_1$ and $x_2$ of $x$, and let $(q_1, f_1)$ be the configuration reached by $E$ at $x_1$ and $(q_2, f_2)$ the configuration reached by $E$ at $x_2$. By induction on $(\star)$ we have trees $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ such that there is a run of $D$ on $\boldsymbol{t}_1 = \boldsymbol{a}_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{t}_2 = \boldsymbol{a}_2 \otimes \boldsymbol{d}_2$ with the appropriate properties. We construct from $\boldsymbol{d}_1$, $\boldsymbol{d}_2$ and the transition of $E$ giving $(q, f)$ a tree $\boldsymbol{d} = d(\boldsymbol{d}_1, \boldsymbol{d}_2)$ such that $D$ also has the expected run on $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$.

For each constraint $\theta$ of the form $C_{u_1} \ominus C_{u_2} \to C_u$ or $C_{u_1} \ominus C_{u_2} \to_1 C_u$, where $u_1$, $u_2$ $u \in {Q_B}^5$, we let $n_\theta$ be the number used by the run of $E$ when using this transition. By induction hypothesis $(\star)$ there are at least $n_\theta$ data values in $\boldsymbol{d}_1$ having $u_1$ as associated $B$-state in $\boldsymbol{a}_1 \otimes \boldsymbol{d}_1$, and $n_\theta$ data values in $\boldsymbol{d}_2$ having $u_2$ as associated $B$-state in $\boldsymbol{a}_2 \otimes \boldsymbol{d}_2$. For showing $(\star)$, we pick $n_\theta$ such data values in each subtrees and call them the data values associated to $\theta$. We do this for all constraints $\theta$ and we choose the associated data values such that they are all distinct. This is possible because of the semantic of the constraints, making sure the counters are big enough and therefore, by $(\star)$, that there is enough data values. The constraints of the form $C_{u_1} \ominus C_{u_2} \to_1 C_u$ where $u$ is a non #-state, for which $n_\theta = 1$, will ensure the propagation of $(\diamond)$. We now apply to $\boldsymbol{d}_2$ a permutation on the data values such that for all $\theta$ the data values associated to $\theta$ in $\boldsymbol{d}_2$ are identified with the ones for $\boldsymbol{d}_1$ and such that all other data values are distinct. In order to simplify the notations we call the resulting tree also $\boldsymbol{d}_2$. We

[5] For the sake of readability, we index the $C$'s with states of $B$ and not integers.

then construct $\boldsymbol{d}$ by adding a root of label $d$ to the $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, where $d$ is chosen according to the guesses made by $E$: $d$ is equal to the data value of the root of $\boldsymbol{d}_1$ if $E$ guessed so, to the data value of the root of $\boldsymbol{d}_2$ if $E$ guessed so, or to an arbitrary data value of $\boldsymbol{d}_1$ or $\boldsymbol{d}_2$ whose associated $B$-states are the one guessed by $E$. We will show in each case that $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ has the desired properties.

**1.** *$E$ guessed that the data value $d$ of the current node is equal to the data value of both its children.*

For this case, the set $\Theta$ contains the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$ for the unique state $p$ such that $(p_1, a', p_2) \to p$ is a transition of $B$, and constraints of the form $C_{v_1} \ominus C_{v_2} \to C_v$ for each transition $(v_1, \#, v_2) \to v$ of $B$ where $v_1$, $v_2$ are #-states.

**Correctness.** Recall the construction of $\boldsymbol{d}$. Notice that the presence of the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$ guarantees that data values of $x$, $x_1$ and $x_2$ are equal in $\boldsymbol{d}$ (recall that $p_1$ and $p_2$ are the root-states of $q_1$ and $q_2$). Let $\boldsymbol{a}'$ be the outcome of the run of $A$ on $\boldsymbol{a}$ as simulated by $E$, and let $\boldsymbol{a}'_1$ and $\boldsymbol{a}'_2$ be its left and right child. Let $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}$. Let $e$ be a data value occurring in $\boldsymbol{d}$.

For $e = d$, the root symbol of $\boldsymbol{t}'[e]{\downarrow_\#}$ is $a'$ and $C_p$ is increased by 1 as expected, thanks to the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_p$. By induction hypothesis ($\diamond$), and since $B$ is deterministic and complete, $f(p) = 1$ and for all other non #-state the corresponding value via $f$ will be 0. Hence $p$ is the new root-state of $q$ and $f$ and ($\diamond$) holds.

For $e \neq d$ we consider 3 subcases. If $e$ occurs in both $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ then $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}_1, \boldsymbol{s}_2)$ for some non empty forests $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Let $v_1$ and $v_2$ be the states reached by $B$ when evaluating $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, *i.e.* they are the $B$-states associated to $e$ in $\boldsymbol{a}'_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{a}'_2 \otimes \boldsymbol{d}_2$. By construction of $\boldsymbol{d}$, there are $n_\theta$ such data values $e$, where $\theta = C_{v_1} \ominus C_{v_2} \to C_v$ for the appropriate $v$. These $n_\theta$ data values will contribute to an increase of $C_v$ by $n_\theta$ as expected.

Assume now that $e$ occurs in $\boldsymbol{d}_1$ but not in $\boldsymbol{d}_2$ (the remaining case being symmetrical). Then $\boldsymbol{t}'[e]$ has the form $\#(\#(\boldsymbol{s}), \#)$ where $\boldsymbol{s}$ is some non empty forest and therefore $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}')$. Hence the state associated to $e$ on $\boldsymbol{t}'$ is the same as the one associated to $e$ on $\boldsymbol{t}_1$. This is consistent with the behavior of $E$ that propagates upward the value of the counter corresponding to this state, after applying the constraints.

Altogether this shows that $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d}$ verifies ($\star$).

**2.** *$E$ guessed that the data value $d$ of the current node is equal to the data value of its left child but different from the data value of its right child. Moreover $E$ guessed that $d$ appears in the right subtree of the current node with $u_2$ as associated $B$-state. Moreover the data value of the right child of the current node also appear in its left subtree with $u_1$ as associated $B$-state. Note that both $u_1$ and $u_2$ must be #-states.*

For is case, the set $\Theta$ contains the constraints $C_{p_1} \ominus C_{u_2} \to_1 C_p$ and $C_{u_1} \ominus C_{p_2} \to_1 C_{p'}$, for the states $p$ and $p'$ such that $(p_1, a', u_2) \to p$ and $(u_1, \#, p_2) \to p'$ are transitions of $B$, and the constraints of the form $C_{v_1} \ominus C_{v_2} \to C_v$ for each transition $(v_1, \#, v_2) \to v$ of $B$ where $v_1$, $v_2$ are #-states.

**Correctness.** Notice that the presence of the constraints $C_{p_1} \ominus C_{u_2} \to_1 C_p$ guarantees that the data values of $x$ and $x_1$ are equal in $\boldsymbol{d}$ and that the presence of the constraint $C_{u_1} \ominus C_{p_2} \to_1 C_{p'}$ guarantees that the data value of $x_2$ also appear in $\boldsymbol{d}_1$. Let $\boldsymbol{a}'$ be the outcome of the run of $A$ on $\boldsymbol{a}$ as simulated by $E$, and let $\boldsymbol{a}'_1$ and $\boldsymbol{a}'_2$ be its left and right child. Let $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}$. Let $d_2$ be the data value at $x_2$. Let $e$ be a data value occurring in $\boldsymbol{d}$.

For $e = d$, the root symbol of $\boldsymbol{t}'[e]{\downarrow_\#}$ is $a'$ and $C_p$ is increased by 1 as expected. By induction hypothesis ($\diamond$), and since $B$ is deterministic and complete, $f(p) = 1$ and for all other non #-state the corresponding value via $f$ will be 0. Hence $p$ is the new root-state of $q$ and $f$.

The case $e = d_2$ is handled similarly with $\theta = C_{u_1} \ominus C_{p_2} \to_1 C_{p'}$.

For $e \neq d$ and $e \neq d_2$, we consider 3 subcases. If $e$ occurs in both $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ then $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}_1, \boldsymbol{s}_2)$ for some non empty forests $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Let $v_1$ and $v_2$ be the states reached by $B$ when evaluating $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, *i.e.* they are the $B$-states associated to $e$ in $\boldsymbol{a}'_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{a}'_2 \otimes \boldsymbol{d}_2$. By construction of $\boldsymbol{d}$, there are $n_\theta$ such data values, where $\theta = C_{v_1} \ominus C_{v_2} \to C_v$. These $n_\theta$ data values will contribute to an increase of $C_v$ by $n_\theta$ as expected.

Assume now that $e$ occurs in $\boldsymbol{d}_1$ but not in $\boldsymbol{d}_2$ (the remaining case being symmetrical). Then $\boldsymbol{t}'[e]$ has the form $\#(\#(\boldsymbol{s}), \#)$ where $\boldsymbol{s}$ is some non empty forest and therefore $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}')$. Hence the state associated to $e$ on $\boldsymbol{t}'$ is the same as the one associated to $e$ on its left child $\boldsymbol{t}'_1$. This is consistent with the behavior of $E$ that propagates upward the value of the counter corresponding to this state, after applying the constraints.

Altogether this shows that $\boldsymbol{a} \otimes \boldsymbol{d}$ has the desired properties.

**3.** *$E$ guessed that the data value $d$ of the current node is equal to the data value of its left child but different from the data value of its right child. Moreover $E$ guessed that $d$ also appear in the right subtree of the current node with $u_2$ as associated $B$-state. Moreover the data value of the right child of the current node does not appear in its left subtree. Note that $u_2$ must be a #-state.*

For this case, the set $\Theta$ contains the constraints $C_{p_1} \ominus C_{u_2} \to_1 C_p$ and $\ominus C_{p_2} \to_1 C_{p'}$ for the state $p$ such that $(p_1, a', u_2) \to p$ and $(\#, p_2) \to p'$ are transitions of $B$, and constraints of the form $C_{v_1} \ominus C_{v_2} \to C_v$ for each transition $(v_1, \#, v_2) \to v$ of $B$ where $v_1$, $v_2$ and $v$ are #-states.

**Correctness.** Notice that the presence of the constraint $C_{p_1} \ominus C_{u_2} \to_1 C_p$ guarantees that the data values of $x$ and $x_1$ are equal in $\boldsymbol{d}$ and that the constraint $\ominus C_{p_2} \to_1 C_{p'}$, together with the absence of constraints containing a non #-state makes sure that the data value of $x_2$ does not appear in $\boldsymbol{d}_1$. Let $\boldsymbol{a}'$ be the outcome of the run of $A$ on $\boldsymbol{a}$ as simulated by $E$ and let $\boldsymbol{a}'_1$ and $\boldsymbol{a}'_2$ be its left and right child. Let $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}$. Let $d_2$ be the data value at $x_2$. Let $e$ be a data value occurring in $\boldsymbol{d}$.

For $e = d$, the root symbol of $\boldsymbol{t}[e]{\downarrow_\#}$ is $a'$ and $C_p$ is increased by 1 as expected thanks to the constraint $C_{p_1} \ominus C_{u_2} \to_1 C_p$. By induction hypothesis ($\diamond$), and since $B$ is deterministic and complete, $f(p) = 1$ and for all other non #-state the corresponding value via $f$ will be 0. Hence $p$ is the new root-state of $q$ and $f$.

The case $e = d_2$ is handled similarly with $\ominus C_{p_2} \to C_{p'}$.

For $e \neq d$ and $e \neq d_2$, we consider 3 subcases. If $e$ occurs in both $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ then $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}_1, \boldsymbol{s}_2)$ for some non empty forests $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. Let $v_1$ and $v_2$ be the states reached by $B$ when evaluating $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, *i.e.* they are the $B$-states associated to $e$ in $\boldsymbol{a}'_1 \otimes \boldsymbol{d}_1$ and $\boldsymbol{a}'_2 \otimes \boldsymbol{d}_2$. By construction of $\boldsymbol{d}$ there are $n_\theta$ such data values, where $\theta = C_{v_1} \ominus C_{v_2} \to C_v$. These $n_\theta$ data values will contribute to an increase of $C_v$ by $n_\theta$ as expected.

Assume now that $e$ occurs in $\boldsymbol{d}_1$ but not in $\boldsymbol{d}_2$ (the remaining case being symmetrical). Then $\boldsymbol{t}'[e]$ has the form $\#(\#(\boldsymbol{s}), \#)$ where $\boldsymbol{s}$ is some non empty forest and therefore $\boldsymbol{t}'[e]{\downarrow_\#}$ has the form $\#(\boldsymbol{s}')$. Hence the state associated to $e$ on $\boldsymbol{t}'$ is the same as the one associated to $e$ on its left child $\boldsymbol{t}'_1$. This is consistent with the behavior of $E$ that propagates upward the value of the corresponding to this state, after applying the constraints.

Altogether this shows that $\boldsymbol{a} \otimes \boldsymbol{d}$ has the desired properties.

**4.** *E guessed that the data value d of the current node is different from the ones of its children but appear in both subtrees with $u_1$ and $u_2$ as associated B-states. Moreover E guessed that the data values of both children of the current node are equal. Note that $u_1$ and $u_2$ must be #-states.*

For this case, the set $\Theta$ contains the constraint $C_{u_1} \ominus C_{u_2} \to_1 C_p$, the constraint $C_{p_1} \ominus C_{p_2} \to_1 C_{p'}$ for the states $p, p'$ such that $(u_1, a', u_2) \to p$ and $(p_1, \#, p_2) \to p'$ are transitions of $B$, and the constraints of the form $C_{v_1} \ominus C_{v_2} \to C_v$ for each transition $(v_1, \#, v_2) \to w$ of $B$ where $v_1$, $v_2$, and $v$ are #-states.

**Correctness.** Recall the construction of $\boldsymbol{d}$. Again the constraints of the form $\to_1$ enforces that the data values of $x$, $x_1$ and $x_2$ matches the guesses of $E$.

The rest of the argument is similar to the previous cases.

**5.** *E guessed that the data value d of the current node is different from the ones of its children but appear in both subtrees with $u_1$ and $u_2$ as associated B-states. Moreover E guessed that the data values of both children of the current node are distinct but appear in other subtree with respective associated B-state $v_1$ and $v_2$. Note that $u_1, u_2, v_1, v_2$ must be #-states.*

For this case, the set $\Theta$ contains the constraints $C_{u_1} \ominus C_{u_2} \to_1 C_p$, $C_{p_1} \ominus C_{v_2} \to_1 C_{p'}$, and $C_{v_1} \ominus C_{p_2} \to_1 C_{p''}$ where the states $p, p', p''$ are such that $(u_1, a', u_2) \to p$, $(p_1, \#, v_2) \to p'$ and $(v_1, \#, p_2) \to p''$ are transitions of $B$, and one constraint $C_{w_1} \ominus C_{w_2} \to C_w$ for each transition $(w_1, \#, w_2) \to w$ of $B$ where $w_1$, $w_2$ are #-states.

**Correctness.** Recall the construction of $\boldsymbol{d}$. Again the constraints of the form $\to_1$ enforces that the data values of $x$, $x_1$ and $x_2$ matches the guesses of $E$.

The rest of the argument is similar to the previous cases.

**6.** *E guessed that the data value d of the current node is different from the ones of its children but appear in both subtrees with $u_1$ and $u_2$ as associated B-states. Moreover it guessed that the data values of the right child of the current node appear in its left subtree with $v_1$ as associated B-state and that the data value of*

*the left child does not appear in the right subtree. Note that $u_1, u_2$ and $v_1$ must be #-states.*

For this case, the set $\Theta$ contains the constraints $C_{u_1} \ominus C_{u_2} \to_1 C_p$, $C_{p_1} \ominus \to C_{p''}$, $C_{v_1} \ominus C_{p_2} \to_1 C_{p'}$ where the states $p, p'$ are such that $(u_1, a', u_2) \to p$, $(p_1, \#) \to p''$ and $(v_1, \#, p_2) \to p'$ are transitions of $B$, and one constraint $C_{w_1} \ominus C_{w_2} \to C_w$ for each transition $(w_1, \#, w_2) \to w$ of $B$ where $w_1$ and $w_2$ are #-states.

**Correctness.** Recall the construction of $\boldsymbol{d}$. Again the constraints of the form $\to_1$ enforces that the data values of $x$, $x_1$ and $x_2$ matches the guesses of $E$.

The rest of the argument is similar to the previous cases.

**7.** *$E$ guessed that the data value $d$ of the current node is different from the ones of its children but appear in both subtrees with $u_1$ and $u_2$ as associated $B$-states. Moreover it guessed that the data values of both children of the current node do not appear elsewhere. Note that $u_1$ and $u_2$ must be #-states*

For this case, the set $\Theta$ contains the constraints $C_{p_1} \ominus \to_1 C_{p'}$, and $\ominus C_{p_2} \to_1 C_{p''}$ where $(p_1, \#) \to p''$ and $(\#, p_2) \to p'$ are transitions of $B$, the constraint $C_{u_1} \ominus C_{u_2} \to_1 C_p$ for the state $p$ such that $(u_1, a', u_2) \to p$ is a transition of $B$, and one constraint $C_{w_1} \ominus C_{w_2} \to C_w$ for each transition $(w_1, \#, w_2) \to w$ of $B$ where $w_1$ and $w_2$ are #-states.

**Correctness.** For the four former cases 4-7, recall the construction of $\boldsymbol{d}$. The constraints of the form $\to_1$ enforces that the data values of $x$, $x_1$ and $x_2$ matches the guesses of $E$. The rest of the argument is similar to the cases 1-3.

**8.** *We omit the symmetric cases.*

## E From EBVASS to $\mathrm{FO}^2(<, +1, \sim)$ (Theorem 3)

**Theorem 3.** The reachability problem for EBVASS reduces to the satisfiability problem for $\mathrm{FO}^2(<, +1, \sim)$.

We show here that the formula $\phi$ constructed in the proof of Theorem 3 in section 4 has the desired property: $\phi$ has a model iff $(p, f_0)$ is reachable by $E$.

*From reachability to models of $\phi$.* Assume that $(p, f_0)$ is reachable and let $\rho$ be a run of $E$ witnessing this fact. Let $\boldsymbol{a}$ be the tree constructed from $\rho$ by concatenating the sequences of encodings of transitions of $\rho$ as explained above. The binary tree $\boldsymbol{a}$ certainly satisfies the "regular" part of $\phi$. We now assign the data values so that the remaining part of $\phi$ is satisfied. This is done in the obvious way: each time a counter $i$ is decremented, as the resulting value is positive, this means that a matching increment was performed before. Similarly, each time a constraint $\theta$ is used in a transition $\mu$, we assign one distinct data value per triple $L_{i_1}, R_{i_2}, T_i$ occurring in the encoding of $\mu$. The formula was constructed to make the resulting tree a model of $\phi$.

*From models of $\phi$ to reachability.* Assume now that $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d} \models \phi$. Unfortunately $\boldsymbol{a}$ could only encode a pseudo-run of $E$. By this we mean that some section may correspond to a pseudo-encoding of an up-transition, instead of an expected real encoding. However, we show that from $\boldsymbol{t}$ we can construct another tree $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}'$ such that $\boldsymbol{t}' \models \phi$ and $\boldsymbol{a}'$ encodes a real run of $E$.

To see this, let us consider a node $x$ of $\boldsymbol{t}$ such that $\boldsymbol{a}(x) = T_\theta$ and let $d = \boldsymbol{d}(x)$. This node $x$ is part of the pseudo-encoding $\boldsymbol{u}$ of some up-transition $\mu$ involving the constraint $\theta = C_{i_1} \ominus C_{i_2} \to C_i$. Let $x_1$ and $x_2$ be two descendants of $x$ such that $\boldsymbol{a}(x_1) = L_\theta$, $\boldsymbol{a}(x_2) = R_\theta$ and $d = \boldsymbol{d}(x_1) = \boldsymbol{d}(x_2)$. The existence of $x_1$ and $x_2$ is guaranteed by $\phi$.

If $x_1$ does not appear in $\boldsymbol{u}$ then we modify the tree $\boldsymbol{t}$ as follows. We cut out $x_1$ and its parent, which must have label $D_{i_1}$ according to the regular constraints, and insert it back in $\boldsymbol{u}$ at the required position. Notice that the two-node segment has been moved up in the tree. The reader can verify that the resulting tree is still a model of $\phi$: the regular conditions remains obviously satisfied. Conditions 1-4 are still valid because the node of label $I_{i_1}$ matching the parent of $y$ was already below the initial position of $y$ and its new position is upward in the tree. Finally conditions 5-8 remain valid by construction.

By symmetry we perform the same operation for $x_2$. Repeating this argument eventually yields a model $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}'$ of $\phi$ such that $\boldsymbol{a}'$ is a correct sequencing of encodings of transitions a $E$. This encoding is actually a real runs because conditions 1-4 of $\phi$ immediately enforces that no counter is ever negative. □

## F Undecidability of DA$^\flat$ (Proposition 1)

**Proposition 1.** Emptiness of DA$^\flat$ is undecidable.

*Proof.* We show that DA$^\flat$ can simulate the Class Automata of [3]. This latter model has an undecidable emptiness problem, already when restricted to data words. It captures indeed the class of languages of words - without data - recognized by counter automata.

We assume given two finite alphabets $\mathbb{A}$ and $\mathbb{B}$, writing the latter *in extenso* as $\mathbb{B} = \{b_1, \ldots, b_n\}$. A class automaton over $\mathbb{A} \times \mathbb{D}$ is a pair $(A, B)$ where $A$ is a letter-to-letter forest transducer from $\mathbb{A}$ into $\mathbb{B}$ and $B$ is a forest automaton taking as input trees in Forests($\mathbb{B} \times \{0, 1\}$). The acceptation of a data forest $\boldsymbol{t} = \boldsymbol{a} \otimes \boldsymbol{d} \in$ Forests($\mathbb{A} \times \mathbb{D}$) by $(A, B)$ is defined like for a DA$^\flat$ except that in the second step, it is required that for all $d$ occurring in $\boldsymbol{d}$, $B$ accepts the forest $\boldsymbol{t}'[\![d]\!]$, where $\boldsymbol{t}'[\![d]\!]$ is defined from $\boldsymbol{t}' = \boldsymbol{a}' \otimes \boldsymbol{d}$ by replacing for every node $x$ of $\boldsymbol{t}'$ its label $\langle a', e\rangle$ by $\langle a', 1\rangle$ if $e = d$ and by $\langle a', 0\rangle$ otherwise.

Given a symbol $a$ and $0 \leq i \leq n$, let $\langle a, d\rangle^i$ be the binary data tree of depth $i$, $a^1 = a$ and $a^{i+1} = a(a^i)$. defined recursively by: $a^0 = \#$ and $a^{i+1} = a(a^i + \#)$. This notation is extended to pairs made of a symbol and a data value $d \in \mathbb{D}$ by adding $d$ at all the nodes: $\langle a, d\rangle^0 = \langle \#, d\rangle$ and $\langle a, d\rangle^{i+1} = \langle a, d\rangle\big(\langle a, d\rangle^i + \langle \#, d\rangle\big)$.

We call *data word* a unary data tree of Trees($\mathbb{B} \times \mathbb{D}$). Let $\mathbb{O}$ be an alphabet containing two symbols $\circ$ and $\#$. We associate to every data word $w \in (\mathbb{B} \times \mathbb{D})^+$

a binary data tree $\boldsymbol{t}_w \in \mathrm{Trees}(\mathbb{O} \times \mathbb{D})$ defined recursively by $\boldsymbol{t}_\varepsilon = \langle \#, 0 \rangle$ ($\varepsilon$ is the empty word), and $\boldsymbol{t}_{\langle b_i, d \rangle w} = \langle \circ, d \rangle (\langle \circ, d \rangle^i, \boldsymbol{t}_w)$.

Let $C = (A, B)$ be a given class automaton computing only on data words of $\mathrm{Trees}(\mathbb{A} \times \mathbb{D})$. It is not difficult to construct a tree automaton $B'$ accepting exactly the set of trees $\boldsymbol{t}_w[d]$ such that $w[\![d]\!]$ is accepted by $B$, for all $d \in \mathbb{D}$. The automaton $B'$ contains the states of $B$ plus some additional states $p_{i,j}$ with $1 \leq 0 \leq n$ and $j = 0$ or 1, such that $p_{i,0}$ and $p_{i,1}$ accept respectively the singletons languages $\{\#^i\}$ and $\{\circ^i\}$, with transitions of the form $\# \to p_{0,0}$, $\# \to p_{0,1}$, and $(p_{i,0}, \#, p_{0,0}) \to p_{i+1,0}$, $(p_{i,1}, \circ, p_{0,0}) \to p_{i+1,1}$ for all $0 \leq i \leq n-1$. We also add to $B'$ a transition $\# \to q_0$, where $q_0$ is the initial state of $B$. Finally, for each transition of $B$ of the form $(q, \langle b_i, 0 \rangle) \to q'$, (respectively $(q, \langle b_i, 1 \rangle) \to q'$), where $q$ and $q'$ are states of $B$, $B'$ contains a transition $(p_{i,0}, \#, q) \to q'$ (respectively $(p_{i,1}, \circ, q) \to q'$).

Defining the set of final states of $B'$ as the final states of $B$, it is easy to verify the equivalence of the emptiness between the languages recognized respectively by the $\mathrm{DA}^\flat$ $(A, B')$ and the class automaton $C = (A, B)$.