Logics for Unordered Trees with Data Constraints on Siblings

Adrien Boiret[1,2], Vincent Hugot[3,2], Joachim Niehren[3,2], and Ralf Treinen[4]

[1] University Lille 1, France
[2] Links (Inria Lille & LIFL, UMR CNRS 8022), France
[3] Inria, France
[4] Univ Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS, F-75205 Paris, France

Abstract. We study counting monadic second-order logics (CMso) for unordered data trees. Our objective is to enhance this logic with data constraints for comparing string data values attached to sibling edges of a data tree. We show that CMso satisfiability becomes undecidable when adding data constraints between siblings that can check the equality of factors of data values. For more restricted data constraints that can only check the equality of prefixes, we show that it becomes decidable, and propose a related automaton model with good complexities. This restricted logic is relevant to applications such as checking well-formedness properties of semi-structured databases and file trees. Our decidability results are obtained by compilation of CMso to automata for unordered trees, where both are enhanced with data constraints in a novel manner.

1 Introduction

Logics and automata for unordered trees were studied in the last twenty years mostly for querying Xml documents [14,5,20] and more recently in the context of NoSql databases [2]. They were already studied earlier, for modeling syntactic structures in computational linguistics [16] and records in programming languages [17,11,12]. In our own work, we also find them relevant to the modeling and static verification of file trees, i.e. structures representing directories, files, their contents etcetera, and their transformations, i.e. programs or scripts moving, deleting, or creating files.

Using unordered trees means expressing and evaluating properties on sets – or multisets – of elements, e.g. the data values of the children at the current position. Naturally, this amounts to counting: for instance in a file tree "*there are at least 2 values that match* *.txt" (where * matches any string), or in a bibliographical database "*there are fewer values* inproceedings *than* book". Where the existing approaches differ is in the expressive power available for that counting; for instance, is it possible to compare two variable quantities – as in the second example – or just one variable quantity and a constant – as in the first. In all cases, however, each element is considered alone, in isolation from its brothers. We previously studied the complexity of decision problems for

```
{ "file.tex" :
  {"\documentclass...":{}},
  "dir" : {
    "x.png" : {"<bin>":{}},
    "y.png" : {"<bin>":{}},
  ...} ... }
```

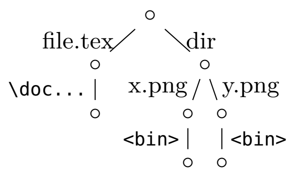

Fig. 1. Unordered trees in JSON format, describing a typical file tree.

automata using various such formalisms as guards for their bottom-up transitions in [3]. The focus was on devising good notions of deterministic machines capable of executing such counting operations, sufficiently expressive but allowing for efficient algorithms. Our present focus, in contrast, is to extend the expressive power of the counting formalisms, while preserving decidability. Since the bottom-up automaton's structure does not play a great role in that, and the yardsticks of expressive power for counting tests are logics, this paper mostly deals directly with second-order logics rather than automata.

Our main goal in this paper is to extend existing formalisms with the ability to express data constraints on unordered data trees, so that each data value may be considered not only in isolation, but also along with sibling values with which it is in relation. Such constraints arise naturally in various circumstances.

By way of example, consider a directory containing LaTeX resources, which may be represented by an edge-labeled tree in the style of Figure 1, given in JSON (JavaScript Object Notation) syntax, where each data value corresponds to a file name or, in the case of leaves, file contents.

Suppose that we want to check whether the contents of a LaTeX repository have been properly compiled, which is to say that for every main LaTeX file – i.e. a file whose name has suffix ".tex", and whose contents begin with "\documentclass" – there exists a corresponding PDF file following the version 1.5 of the standard. To express this property, sibling data values – here representing files in the same directory – are put in relation by

$$\theta_{\text{tex2pdf}} = \{\, (w\texttt{".tex"}, w\texttt{".pdf"}) \mid w \text{ is a word} \,\} \ . \tag{1}$$

Schematically, we express constraints of the form "any value d whose subtree satisfies some property P has a brother $d' = \theta_{\text{tex2pdf}}(d)$ whose subtree satisfies another property P'".

We need to integrate that kind of data constraints in existing formalisms for unordered trees; the two yardsticks of expressive power that have emerged in the literature are the extensions of weak monadic second-order logic (MSO) by horizontal Presburger constraints [14], and by the weaker, but more tractable, counting constraints [8], capable of expressing that the cardinality of a set variable is *less than m* or *equal to m modulo n*, but not of comparing the cardinalities of two set variables directly, unlike Presburger logic. We choose MSO with counting constraints as our starting point, which we write CMSO. We denote by $\mathit{\Gamma}(\mathit{\Theta})$ and CMSO($\mathit{\Theta}$) the extensions of counting constraints and CMSO, respectively, with tests on siblings for a certain class $\mathit{\Theta}$ of binary relations on data words. Provided that this class contains the relation θ_{tex2pdf} defined above, our example property that, everywhere in the file tree, every TEX file has a corresponding PDF is expressed by the CMSO($\{\theta_{\text{tex2pdf}}\}$) formula

$$\forall x \,.\, x \in (\#(*\texttt{".tex"} \;\wedge\; X_{\text{doc}} \;\wedge\; \neg\theta_{\text{tex2pdf}}.X_{\text{pdf15}}) = 0) \tag{2}$$

where X_{doc} and X_{pdf15} are free set variables assumed to contain the nodes satisfying the "main TEX file" and "valid PDF" properties. Intuitively: "all nodes in the file tree are among the nodes such that the number of their children whose label matches *".tex", which are main TEX documents, and for which there does not exist a corresponding *".pdf" sibling that is a PDF version 1.5, is zero." We shall give the full, closed formula at the end of Sec. $3_{[\text{p5}]}$.

Note that, even for *ordered* data words and in the case of equality tests, – simpler than even the suffix correspondences exemplified by θ_{tex2pdf} – satisfiability, and the emptiness problem in the case of automata, become rapidly intractable or undecidable. This has been studied for register automata, first-order logic, and XPath, [4,9], among others. For instance, satisfiability of $FO^2(=, +1, <)$, i.e. first-order logic with two variables and *successor* and *linear order* relations, while decidable, is not known to be primitive recursive [4]. Unorderedness simplifies matters in this case.

Nevertheless, the choice of the class of string relations $\mathit{\Theta}$ to which we have access in our constraints greatly influences the complexity and decidability of the counting constraints using them. We have found that even relatively conservative choices of $\mathit{\Theta}$ entail undecidability: merely allowing the replacement of factors of up to three letters, or the addition and deletion of suffixes and prefixes of one letter, suffices. However, we exhibit a relatively large class which *is* decidable, and capable of expressing that the prefixes of two data values are the same, or even in the same regular language, while the suffixes belong to two different languages; this largely covers our envisioned applications. We have also found further restrictions for which the complexities become more reasonable.

Outline:
After a few preliminaries, **Section 3** introduces the logic CMSO($\mathit{\Theta}$), which is CMSO extended with the ability to put an edge's data value in relation with one of its siblings', the string relation being a member of $\mathit{\Theta}$. In **Section 4**, we show that if relations allow both prefix and suffix manipulations, even restricted to addition or removal of a single letter, CMSO($\mathit{\Theta}$) becomes undecidable. After recalling the logic WS*k*S, which covers a large class of suffix-only manipulations, **Section 5** shows that CMSO($\mathit{\Theta}_{\text{WS}k\text{S}}$), where allowed relations are WS*k*S-definable relations, is decidable in non-elementary time, by translating it into automata with horizontal tests in WS*k*S. **Section 6** presents an algorithm that decides emptiness for the automaton model equivalent to a fragment of the logic where string relations

are limited to disjoint suffix replacements. Its complexity is NExpTime – or PSpace if the automaton is deterministic. **Section 7** concludes and hints at possible extensions and different ways of tackling the problem.

2 MSO and Counting Constraints

We recall the definition of MSO and of counting constraints. As models we restrict ourselves to data trees, even though general graph structures could be chosen.

Data Trees. A *data alphabet* is a finite set $\mathbb{A}$. A *data value* over $\mathbb{A}$ is a string in $\mathbb{A}^*$. The *trees* under consideration are finite, unordered, unranked trees whose edges are labeled by data values in $\mathbb{A}^*$. Formally, a tree t is a multiset $\{\!|\, (d_1, t_1), \ldots, (d_n, t_n) \,|\!\}$ where $d_1, \ldots, d_n \in \mathbb{A}^*$ and $t_1, \ldots, t_n$ are trees. d_i is the label of the edge leading into the subtree t_i. For instance, the tree of Fig 1 is $\{\!|\, (\texttt{"file.tex"}, \{\!|\, (\texttt{"\textbackslash doc..."}, \{\!|\ |\!\}) \,|\!\}), (\texttt{"dir"}, \ldots) \,|\!\}$. To simplify the formalisation, we shall not manipulate edges as distinct objects, but instead see an edge label as a property of the node into which the edge leads. Thus we assimilate t to a structure $\langle \mathsf{V}_t, \ell_t, \downarrow_t \rangle$, where V_t is the set of nodes of t, $\ell_t(v)$ is the data value labeling the edge leading into the node v – undefined for the root node, – and $v \downarrow_t v'$ holds if v' is a child of v. For our convenience, we also define the "sibling-or-self" relation: $v \curlywedge_t v' \;\; \Leftrightarrow \;\; \exists v'' \,.\, v'' \downarrow_t v \;\wedge\; v'' \downarrow_t v'$. By extension of the language of ranked trees, we use the word *arity* to refer to either the multiset of outgoing edge labels of a node, or the set of outgoing edges.

MSO. Let $\mathbb{A}$ be a data alphabet and $\mathcal{X}$ a countable set of variables of two types, node variables and set variables. A variable assignment I into some tree t will map any node variable $x \in \mathcal{X}$ to a node $I(x) \in \mathsf{V}_t$ and any set variable $X \in \mathcal{X}$ to a set of nodes $I(X) \subseteq \mathsf{V}_t$.

As a parameter of our logic we assume a set Ψ of formulæ called *node selectors*, which may contain letters from $\mathbb{A}$ and variables from $\mathcal{X}$. The only assumption we make is that any node selector $\psi \in \Psi$ defines for any tree t and variable assignment I into t a set of nodes $[\![\psi]\!]_{t,I} \subseteq \mathsf{V}_t$. For instance, we could choose $\Psi = \Psi_0 = \{\pi \mid \pi \text{ regular expression over } \mathbb{A}\} \cup \{\downarrow x \mid x \in \mathcal{X} \text{ node variable}\}$ such that $[\![\pi]\!]_{t,I} = \{v \mid \ell_t(v) \text{ matches } \pi\}$ is set of all nodes whose incoming edge is labeled by a word in $\mathbb{A}^*$ that matches regular expression π, and $[\![\downarrow x]\!]_{t,I} = \{v \mid v \downarrow_t I(x)\}$ is the set of nodes of which $I(x)$ is a child. Or else, we could also choose $\Psi = \Psi_0 \cup \{\downarrow X \mid X \in \mathcal{X} \text{ set variable}\}$, where a formula $\downarrow X$ requires that some child belongs to $I(X)$. The formulæ of MSO over Ψ are:

$$\xi \in \text{Mso}(\Psi) \;\; ::= \;\; x \in \psi \;\;|\;\; x \in X \;\;|\;\; \exists x \,.\, \xi \;\;|\;\; \exists X \,.\, \xi \;\;|\;\; \xi \wedge \xi \;|\, \neg\xi \quad ,$$

were $\psi \in \Psi$. Whether a formula is true for a given tree t and variables assignment I into t is defined as follows:

$$\begin{array}{ll} t, I \models x \in \psi \Leftrightarrow I(x) \in [\![\psi]\!]_{t,I} & t, I \models \xi \wedge \xi' \Leftrightarrow t, I \models \xi \text{ and } t, I \models \xi' \\ t, I \models x \in X \Leftrightarrow I(x) \in I(X) & t, I \models \neg\xi \Leftrightarrow \text{not } t, I \models \xi \\ \multicolumn{2}{l}{t, I \models \exists x \,.\, \xi \Leftrightarrow t, I[x \mapsto v] \models \xi \text{ for some } v \in \mathsf{V}_t} \\ \multicolumn{2}{l}{t, I \models \exists X \,.\, \xi \Leftrightarrow t, I[X \mapsto V] \models \xi \text{ for some finite } V \subseteq \mathsf{V}_t} \end{array}$$

As syntactic sugar, we will freely use the usual additional logical connectives and set comparisons that can be easily encoded, i.e. formulæ $\forall x.\xi$, $\forall X.\xi$, $\xi \Leftrightarrow \xi'$, $\xi \Rightarrow \xi'$, as well as $X \subseteq X'$, $X = \psi$, and $\psi = \varnothing$.

Children Counting Constraints. A children counting constraint selects a node of a tree by testing the number of its children satisfying some property. Which properties can be tested is defined by the parameter Φ of node selectors. As before, we use as parameter a set of node selectors Φ such that $[\![\phi]\!]_{t,I} \subseteq \mathsf{V}_t$ is defined for all $\phi \in \Phi$, and which may contain variables in $\mathcal{X}$ and letters in $\mathbb{A}$. For instance, we could chose $\Phi = \{\pi \mid \pi \text{ regular expression over } \mathbb{A}\} \cup \mathcal{X}$. A counting constraint over Φ is a formula with the following syntax, where $\phi \in \Phi$ and n, m are natural numbers including 0:

$$\gamma \in \Gamma(\Phi) \quad ::= \quad \#\phi \leqslant n \quad | \quad \#\phi \equiv_m n \quad | \quad \gamma \wedge \gamma \quad | \quad \neg\gamma \quad .$$

The first two kinds of formulæ can test whether the number of children satisfying ϕ is less or equal to n or equal to n modulo m. Note that we *cannot* write $\#\phi \leqslant \#\phi'$, which would lead to the richer class of Presburger formulæ.

Any counting constraint γ defines a set of nodes $[\![\gamma]\!]_{t,I}$ for any variables assignment I to t, so counting constraints themselves can be used as node selectors:

$$\begin{array}{rcl} [\![\#\phi \leqslant n]\!]_{t,I} & = & \{\, v \in \mathsf{V}_t \mid \ \mathrm{Card}(\{\, v' \mid v \downarrow_t v' \ \wedge \ v' \in [\![\phi]\!]_{t,I} \,\}) \leqslant n \,\} \;, \\ [\![\#\phi \equiv_m n]\!]_{t,I} & = & \{\, v \in \mathsf{V}_t \mid \ \mathrm{Card}(\{\, v' \mid v \downarrow_t v' \ \wedge \ v' \in [\![\phi]\!]_{t,I} \,\}) \equiv_m n \,\} \;, \\ [\![\gamma \wedge \gamma']\!]_{t,I} & = & [\![\gamma]\!]_{t,l} \cap [\![\gamma']\!]_{t,l} \;, \qquad [\![\neg\gamma]\!]_{t,I} = \mathsf{V}_t \setminus [\![\gamma]\!]_{t,l} \;. \end{array}$$

Note that we can define $\#\phi \geqslant n$ as syntactic sugar for $\neg(\#\phi \leqslant n-1)$, and $\#\phi = n$ as syntactic sugar for $\#\phi \geqslant n \ \wedge \ \#\phi \leqslant n$.

3 Counting MSO for Data Trees: CMso(Θ)

We now introduce counting MSO for data trees with comparisons of sibling data values. Which precise comparisons are permitted is a parameter of the logic.

As before we assume a set of variables $\mathcal{X}$ and a data alphabet $\mathbb{A}$. In addition, we fix a set Θ of binary relations on $\mathbb{A}^*$ that are called string comparisons. We then define a set of node selectors with regular expressions for matching data values and comparisons of sibling data values from Θ. Such a node selector has the following syntax where $\theta \in \Theta$, π is a regular expression over $\mathbb{A}$, and $x, X \in \mathcal{X}$:

$$\begin{array}{rcll} \phi \in \Phi_{\text{rel}}(\Theta) & ::= & \pi & \text{incoming edge label matches } \pi, \\ & & \mid x \mid X & \text{equal to } x \text{ or member of } X, \\ & & \mid \theta.\phi & \exists \text{ sibling satisfying } \phi \text{ with labels related by } \theta, \\ & & \mid \phi \wedge \phi \mid \neg\phi & \text{conjunction and negation.} \end{array}$$

The sets of selected nodes are defined as follows for formula $\phi \in \Phi_{\text{rel}}$, any tree t and variable assignment I into t:

$$\begin{array}{ll} [\![\pi]\!]_{t,I} = \{v \mid \ell_t(v) \text{ matches } \pi\} & [\![\phi \wedge \phi']\!]_{t,I} = [\![\phi]\!]_{t,I} \cap [\![\phi']\!]_{t,I} \\ [\![x]\!]_{t,I} = \{I(x)\} & [\![\neg\phi]\!]_{t,I} = \mathsf{V}_t \setminus [\![\phi]\!]_{t,I} \\ [\![X]\!]_{t,I} = I(X) & \\ \multicolumn{2}{l}{[\![\theta.\phi]\!]_{t,I} = \{v \mid \exists v' \,.\, v \curlywedgedownarrow_t v' \;\wedge\; (\ell_t(v), \ell_t(v')) \in \theta \;\wedge\; v' \in [\![\phi]\!]_{t,I}\}} \end{array}$$

In particular, a node selector $\theta.\phi$ selects all nodes that have a sibling-or-self, so that the data values of these two nodes satisfy comparison θ.

Definition 1. *We define the children counting contraints for data trees with comparisons of data values $\Gamma(\Theta)$ by $\Gamma(\Phi_{rel}(\Theta))$ and the counting MSO for data trees with comparison of sibling data values* CMSO(Θ) *by* MSO$(\Gamma(\Theta))$.

Note that the childhood $x \downarrow x'$ can be defined in CMSO(Θ) by $x \in (\#x' = 1)$ independently of the choice of Θ. Hence, sibling-or-self contraints $x \curlywedgedownarrow x'$ can also be defined by $\exists x''. \, (x'' \downarrow x \wedge x'' \downarrow x')$ for any Θ. The elements of Θ intervene only if one wants to compare the data values of sibling nodes.

Example 1. Recall the TEX compilation example of equation $(2)_{[p3]}$ and its free variables. There remains to bind X_{doc} and X_{pdf15} to the relevant sets of nodes in a closed formula. A TEX main document (resp. a valid PDF version 1.5) is represented by a node with a single outgoing edge, whose label is prefixed by "\documentclass" (resp. "%PDF-1.5"), leading to a leaf. Thus the closed CMSO$(\{\theta_{\text{tex2pdf}}\})$ formula:

$$\begin{array}{l} \exists X_{\text{leaf}} \,.\, \exists X_{\text{doc}} \,.\, \exists X_{\text{pdf15}} \,. \quad X_{\text{leaf}} \;=\; (\#(*) = 0) \\ \quad \wedge\; X_{\text{doc}} \;=\; (\#(*) = 1 \;\wedge\; \#(\texttt{"\textbackslash documentclass"} * \;\wedge\; X_{\text{leaf}}) = 1) \\ \quad \wedge\; X_{\text{pdf15}} \;=\; (\#(*) = 1 \;\wedge\; \#(\texttt{"\%PDF-1.5"} * \;\wedge\; X_{\text{leaf}}) = 1) \\ \qquad\qquad\qquad \wedge\; \forall x \,.\, x \in (\#(*\texttt{".tex"} \;\wedge\; X_{\text{doc}} \;\wedge\; \neg\theta_{\text{tex2pdf}}.X_{\text{pdf15}}) = 0) \,. \end{array}$$

Example 2. Another useful thing to require of a data tree is the *feature tree* property, stating that no two sibling edges may share the same label. This property can be used to specify files systems, since one needs to state that no two files in the same directory have the same name. Taking θ_{id} as the identity relation, we can define feature trees in CMSO$(\{\theta_{\text{id}}\})$ as follows:

$$\forall x \,.\, (\#(x \;\wedge\; \theta_{\text{id}}.\neg x) \geqslant 1) = \varnothing \,.$$

Example 3. Consider now a transformation θ_{bck}, which to w associates w".bck", thus relating a file's name to that of its automatic backup. Suppose that the system can back up a backup, and so on, up to a certain point, and we need to check that this bound is not overstepped. That is to say, given $n \in \mathbb{N}$, we want to write a formula ξ_n enforcing that there is no chain of backups of length greater than n. Suppose we had a least-fixed point operator μ among our child-selectors, following the syntax – $\mu X.\phi$ – and semantics of μ-calculus. Then we could write ξ_n in CMso($\{\theta_{\text{bck}}\}$):

$$\forall x \;.\; (\#\mu X.(x \;\vee\; \theta_{\text{bck}}.X) > n+1) = \varnothing \;.$$

$\mu X.(x \vee \theta_{\text{bck}}.X)$ intuitively captures the set of nodes related to x by successive iterations of θ_{bck}; we can do the same thing without needing μ by explicitly binding a set variable Y to the least fixpoint of $x \vee \theta_{\text{bck}}.X$, wrt. X:

$$\begin{aligned}&\forall x \;.\; \exists Y \;.\; (\#((x \vee \theta_{\text{bck}}.Y) \wedge \neg Y) \geqslant 1) = \varnothing \\ &\quad\wedge\; \nexists Y' \;.\; Y' \subseteq Y \wedge (\#((x \vee \theta_{\text{bck}}.Y') \wedge \neg Y') \geqslant 1) = \varnothing \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\wedge\; \forall x \;.\; [\#Y > n+1] = \varnothing \;.\end{aligned}$$

The first line establishes Y as a fixed point, as it means that there are no nodes with a child satisfying $x \vee \theta_{\text{bck}}.Y$ but not Y. The second line states that there is no smaller fixpoint than Y. This encoding can be generalised to any use of μ.

4 Undecidable Instances of CMso(Θ)

In this section, we exhibit conditions on the expressive power of the class of data constraints Θ sufficient to render satisfiability for $\Gamma(\Theta)$, and therefore for CMso(Θ), undecidable. As we shall see, not much is needed. Even merely allowing Θ to express the addition or removal of a single letter at the beginning or end of a word is enough; the argument developed in the next theorem is that even this is sufficient to encode the solution of the Post Correspondence Problem.

Theorem 1. *Let Θ_1 be the set of string relations of the forms $w \mapsto wa$, $w \mapsto aw$, $wa \mapsto w$, or $aw \mapsto w$, with $a \in \mathbb{A}, w \in \mathbb{A}^*$. Then* CMso($\Theta_1$) *is undecidable.*

Proof. We reduce PCP, with input dominoes $[\begin{smallmatrix}u_1\\v_1\end{smallmatrix}], \dots, [\begin{smallmatrix}u_n\\v_n\end{smallmatrix}]$. Let us write the relations in Θ_1 as θ_{+a}, θ_{a+}, θ_{-a}, and θ_{a-}, respectively. Given a word $w = a_1 \dots a_m$, by abuse of notation we abbreviate $\theta_{+a_1}.\dots.\theta_{+a_m}.\phi$ into $\theta_{+w}.\phi$. Although Θ_1 is not closed by composition, this construction enables us to pretend that it is – the difference is that it requires the existence of siblings for each intermediate step, which does not affect us. $\theta_{w-}.\phi$ is defined likewise. $\theta_{a_m+}.\dots.\theta_{a_1+}.\phi$ is written $\theta_{w+}.\phi$, and likewise for $\theta_{-w}.\phi$. Let $\$_1, \$_2 \in \mathbb{A}$ be symbols not appearing in any domino, serving as markers for the first and the second phase of the construction. The mirror of u is written $\overline{u}$. The operation for "placing domino i around previous dominoes" is defined as $\theta_i.\phi \equiv \theta_{\$_1-}.\theta_{\overline{u_i}+}.\theta_{+v_i}.\theta_{\$_1+}.\phi$; "accepting dominoes" is $\theta_{\mathrm{acc}}.\phi \equiv \theta_{\$_1-}.(*_1 \wedge \theta_{\$_2+}.\phi)$, where $*_1$ matches any string of length $\geqslant 1$, to avoid the empty sequence as a trivial solution; "reading a on both ends" is $\theta_a.\phi \equiv \theta_{\$_2-}.\theta_{a-}.\theta_{-a}.\theta_{\$_2+}.\phi$. Abbreviating $\theta.*$ or $\theta.\mathit{true}$ into simply θ, consider now the formula $\gamma \in \Gamma(\Theta_1) =$

$$\begin{aligned}&\#\$_1 = 1 \;\wedge\; \#\$_2 = 1 \;\wedge\\ &\qquad \#(\$_1* \wedge\neg(\theta_1 \vee \cdots \vee \theta_n \vee \theta_{\mathrm{acc}})) = 0 \quad\wedge\quad \#(\$_2* \wedge\neg(\textstyle\bigvee_{a\neq \$_1,\$_2}\theta_a)) = 1\ .\end{aligned}$$

It is satisfiable iff there is a tree whose arity contains $\$_1$, $\$_2$, and such that every label beginning with $\$_1$ (i.e. phase one) has a sibling (along with the intermediate siblings) obtained either by placing some domino so that u_i mirrors v_i, staying in phase one, or by moving to phase two. At this point, a label is of the form $\$_2\overline{u_{i_k}}\dots\overline{u_{i_1}}v_{i_1}\dots v_{i_k}$. Furthermore, all but one label beginning with $\$_2$ (i.e. all but $\$_2$) have a sibling obtained by removing the same letter at the beginning and the end; all letters must be read until only $\$_2$ remains. Thus, γ is satisfiable iff there are $i_1,\dots,i_k$ such that $u_{i_1}\dots u_{i_k} = v_{i_1}\dots v_{i_k}$. This shows that $\Gamma(\Theta_1)$ is undecidable. This carries over to CMso(Θ_1): consider the formula $\exists x\ .\ x \in \gamma$. $\square$

5 Satisfiability of CMso($\Theta_{\mathrm{WS}k\mathrm{S}}$) is Decidable

We shall now see that, in spite of the bleak picture painted by the previous section, Θ *can* be made rather large and useful without forgoing decidability. Indeed, the most frequent operation in applications, illustrated in particular by the TEX example $(1)_{[\mathrm{p}2]}$, is suffix replacement. The property that we really need is thus decidability of satisfiability for CMso(Θ_{suffix}), where the relations of Θ_{suffix} are of the form $\theta_{u,u'} = \{\,(wu, wu') \mid w \in \mathbb{A}^*\,\}$, for $u, u' \in \mathbb{A}^*$. We show decidability for a class that is actually more general: WSkS-definable relations.

The well-known logic Weak Monadic Second-Order Logic with k Successors (WSkS) [6], for any $k \geqslant 1$, is based on first-order variables z, and second-order variables Z. Terms τ and formulæ ω of this logic are defined by

$$\begin{array}{rcll}\tau & ::= & \epsilon \mid z \mid \tau i & \qquad\qquad\qquad 1 \leqslant i \leqslant k\\ \omega & ::= & \tau = \tau \mid \tau \in Z \mid \omega \wedge \omega \mid \neg\omega \mid \exists z\ .\ \omega \mid \exists Z\ .\ \omega & \end{array}$$

First-order variables range over words in $\{\,1,\dots,k\,\}^*$, and second-order variables range over finite subsets of $\{\,1,\dots,k\,\}^*$. The constant ϵ denotes the empty word, and each of the functions i, written in postfix notation, denotes appending the symbol i at the end of a word. Validity and satisfiability of formulæ in WSkS are decidable [19], even though with a non-elementary complexity [18].

Some useful relations expressible in WSkS are $z \leqslant_{\mathrm{pref}} z'$ (prefix partial order on words), $z \leqslant_{\mathrm{lex}} z'$ (lexicographic total order on words), $z \in \pi$ for any regular expression π, $Z \subseteq Z'$, $Z = Z' \cup Z''$, $Z = Z' \cap Z''$, $Z = \overline{Z'}$ (complement), $Z = \varnothing$, $|Z| \equiv_n m$ for any constants n, m. Most of those are shown in [7, p88].

The unary predicates on words definable in WSkS are precisely the regular sets [13,10]. A binary relation $R \subseteq \{\,1,\ldots,k\,\}^* \times \{\,1,\ldots,k\,\}^*$ is called *special* if it is of the form $\{\,(ab, ac) \mid a \in L,\ b \in M,\ c \in N\,\}$ for some regular sets L, M, and N. A binary relation on words is definable in WSkS iff it is a finite union of special relations [10]. Some relations which are known *not* to be expressible in WSkS are $z = z'z''$, $z = iz'$, z is a suffix of z', z and z' have the same length, Z and Z' have the same cardinality. Let us note that what is definable largely includes the kinds of suffix manipulations which we need for applications and, conversely, that the dangerous properties highlighted in the previous section are not expressible: one cannot manipulate suffixes and prefixes at the same time.

Let $\Theta_{\mathrm{WS}k\mathrm{S}}$ be the set of WSkS-definable relations, with the letters of $\mathbb{A}$ taken as successor functions, along with a fresh letter \$; we sketch the proof of decidability of CMso($\Theta_{\mathrm{WS}k\mathrm{S}}$). Child-selectors ϕ and counting constraints ψ are encoded into WSkS, and thus shown decidable. The Mso layer can then be translated into automata, yielding a model of automata for unordered trees as in [3], for which the emptiness problem is known to be decidable under certain conditions, which are here satisfied.

We encode multisets A of edge labels w as sets of WSkS strings, accounting for multiplicities $A(w)$ by appending different numbers of \$ to ws to differentiate them. Let t be a tree and A^t_v the arity – the multiset of labels – of node v; the encoding of A^t_v is denoted by $\overline{A^t_v}$ and that of v by $\overline{v}$, such that

$$\overline{A^t_v} = \left\{\, w\$^k \;\middle|\; 1 \leqslant k \leqslant A^t_v(w) \,\right\} = \left\{\, \overline{v'} \;\middle|\; v \downarrow_t v' \,\right\} ,$$

where $\overline{v'} = \ell_t(v')\$^i$ for some i. Note that all children sharing the same label must get a different i; while there are several valid encodings depending on that assignment, we simply choose one, indifferently. Taking $\overline{X}$ as fresh WSkS set variables, this encoding extends to interpretations in the obvious way. We can now encode any child-selector ϕ as a WSkS formula $\overline{\phi}$ with free variables z, Z (standing for the current node and its arity), such that for any tree t, interpretation I, and nodes $v' \downarrow_t v$:

$$t, I, v \models \phi \quad\iff\quad \overline{I}[z \mapsto \overline{v}, Z \mapsto \overline{A^t_{v'}}] \models \overline{\phi}\ .$$

Our building blocks are: (1) $z \models \pi$, where π is a regular expression, which is known to be WSkS-expressible, (2) $z\theta z'$ is expressible by definition, since θ is a WSkS-expressible relation, and (3) $z - \$$, which removes all the \$ at the end of the word, testing its well-formedness at the same time it restitutes the edge-label, and is encoded as

$$z' = z - \$ \quad\equiv\quad z'\$ \leqslant_{\mathrm{pref}} z \quad\wedge\quad z' \models \mathbb{A}^*\ .$$

Using this, we have the following encodings:

$$\begin{aligned} \overline{\pi} \quad&\equiv\quad (z - \$) \models \pi, \qquad \overline{X} \quad\equiv\quad z \in \overline{X}, \\ \overline{\theta.\phi} \quad&\equiv\quad \exists z' \in Z\,.\,(z - \$)\theta(z' - \$) \;\wedge\; \overline{\phi}[z \leftarrow z']\ . \end{aligned}$$

There remains to handle counting constraints ψ, which is simply a matter of showing that WSkS can encode the primitives $|Z| \leqslant m$ – which is easy – and $|Z| \equiv_n m$ – which rests on a total order such as the lexicographic one, and on the idea of affecting each element in turn to a second-order variable corresponding to the value of the modulo. (Note that the same cannot be said of Presburger logic's $|X| = |Y|$ tests, which are not expressible in WSkS, and whose addition would make it undecidable.) With this done, all decidability results for WSkS carry over to $\mathit{\Gamma}(\mathit{\Theta}_{\mathrm{WS}k\mathrm{S}})$; in particular:

Lemma 1. *Satisfiability of $\mathit{\Gamma}(\mathit{\Theta}_{WSkS})$ is decidable.*

There now remains to deal with the Mso layer; it could be encoded in WSkS as well (as it is a second order logic with sufficient expressive power), but it is simpler to take an automaton-based viewpoint, similar to [15,3] (with the addition of θs). We summarise the model of automata for our unordered trees, AUT($\mathit{\Theta}$), as bottom-up automata with rules $\psi \to q$, where ψ are formulæ of $\mathit{\Gamma}(\mathit{\Theta})$ whose child-selectors have an additional test q determining whether a child node has been evaluated in q previously (this corresponds to an "X_q" test). A tree language L is said to be CMso($\mathit{\Theta}$)-definable if there exists a closed formula $\xi_L \in$ CMso($\mathit{\Theta}$) such that $L = \{\, t \mid t \models \xi_L \,\}$. Through straightforward adaptations of the usual encodings [19,7], and further noting that AUT($\mathit{\Theta}$) are effectively closed by all boolean operations, we obtain:

Lemma 2. *A set of trees is* CMso($\mathit{\Theta}$)*-definable iff it is accepted by an* AUT($\mathit{\Theta}$).

Of course, this result is constructive, and we can then adapt the usual reachability algorithm: provided that $\mathit{\Gamma}(\mathit{\Theta})$ is decidable, so is emptiness for AUT($\mathit{\Theta}$), and, in turn, so is CMso($\mathit{\Theta}$). In particular:

Theorem 2. *Satisfiability of* CMso($\mathit{\Theta}_{WSkS}$) *is decidable.*

6 More Efficient Fragments

We can further gain in efficiency by further restricting the θ relation. To this end, we consider mutually exclusive suffix replacement: we pick a set of suffixes $L = \{w_1, \ldots, w_n\}$ such that w_i is never a suffix of another w_j. Let $\mathit{\Theta}_L$ be the set of string relations θ_{w_i,w_j} linking uw_i to uw_j, we denote $\mathit{\Gamma}_{\mathrm{suf}L}$ the counting formulæ of $\mathit{\Gamma}(\mathit{\Theta}_L)$, with the additional restriction that regular expressions testing labels are of the form $\mathbb{A}^* \cdot w_i$. We use a small-model argument to find an efficient algorithm for satisfiability. We will later use this logic in bottom-up automata of AUT($\mathit{\Theta}_L$) as we did in Part 5.

We consider that our arities are already annotated by set variables $X \in \mathcal{X}$. These variables will later correspond to state labelings of an automaton of AUT($\mathit{\Theta}$). If we consider vertically deterministic automata of $\text{AUT}(\Theta)$ [3], where each tree is evaluated in at most one state, the variables X are mutually exclusive. By restricting ourselves to mutually exclusive suffixes, we only need to consider the edges labeled in uL, i.e. the *orbit* of uw_i under the action of all θ_{w_i,w_j}. This allows us to guessing a valid arity for $\phi \in \Gamma_{\text{suf}L}$ orbit by orbit. All we need then is a small-model theorem: if $\#\phi \leqslant n$ appears in a formula ψ, we need to keep track of how many elements are selected by ϕ in a counter that stops at n. if $\#\phi \equiv_m n$ appears in ψ, we need to keep track of how many elements are selected by ϕ in a counter modulo m. This leads to an exponential number of configurations, which means that, if ψ is satisfiable, then we can find a solution using an exponential number of orbits of exponential size. We finally get:

Lemma 3. *The satisfiability problem for an arity formula of Γ_{sufL} is decidable in* NExpTime. *Furthermore, if the variables X are mutually exclusive, the satisfiability problem for an arity formula of Γ_{sufL} is decidable in* PSpace.

We can then use the techniques of [15,3], to extend our results to a class $\text{AUT}(\Theta_L)$ of bottom-up automata with rules $\psi \to q$, where ψ are formulæ of $\Gamma_{\text{suf}L}$.

Theorem 3. *The emptiness problem for automata in $\text{AUT}(\Theta_L)$ is decidable in* NExpTime. *Furthermore, for deterministic automata of $\text{AUT}(\Theta_L)$, the emptiness problem is decidable in* PSpace.

7 Conclusions and Future Works

We have introduced the logic $\text{CMso}(\Theta_{\text{WS}k\text{S}})$ on unordered data trees. It is an extension of CMso to data trees, where tests on a given child may include enforcing the existence of a sibling whose label is in relation with that child's own label, the relation being WSkS-definable. That logic's expressive power is largely sufficient for concrete applications, such as the verification of common constraints on file trees, which usually involve suffix manipulations, largely captured by WSkS. We have shown that satisfiability for $\text{CMso}(\Theta_{\text{WS}k\text{S}})$ is decidable. However, we have also shown that any attempt to allow additional data relations for both prefix *and* suffix manipulations, even of the simplest kind, would render the logic undecidable. We have also studied the complexity of the emptiness tests for automata where horizontal counting constraints are restricted to relations that only involve disjoint suffixes, and shown that the test is then NExpTime for alternating automata, and only PSpace for deterministic automata.

There are two main ways in which this work can be extended. One is to find more expressive string relations for which the logic remains decidable; our undecidability results indicate that such an extension may not be very natural. Another is to extend the reach of the string relation from merely the set of siblings to something larger. In a first step towards that, the proof of Thm 3 can be extended to support equality constraints between brother subtrees without changing the NExpTime complexity. Another promising direction is the use of Monadic Datalog on data trees [1], which is capable of expressing relations not only with siblings but also with parents, cousins etcetera, and for which efficient algorithms are known.

References

1. Abiteboul, S., Bourhis, P., Muscholl, A., Wu, Z.: Recursive queries on trees and data trees. In: ICDT. pp. 93–104. ACM (2013)
2. Benzaken, V., Castagna, G., Nguyen, K., Siméon, J.: Static and dynamic semantics of NoSQL languages. In: POPL. pp. 101–114. ACM (2013)
3. Boiret, A., Hugot, V., Niehren, J., Treinen, R.: Deterministic Automata for Unordered Trees. In: GandALF. Verona, Italy (Sep 2014)
4. Bojanczyk, M., David, C., Muscholl, A., Schwentick, T., Segoufin, L.: Two-variable logic on data words. ACM Trans. Comput. Log. 12(4), 27 (2011)
5. Boneva, I., Talbot, J.M.: Automata and logics for unranked and unordered trees. In: RTA. LNCS, vol. 3467, pp. 500–515. Springer Verlag (2005)
6. Büchi, J.R.: Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly 6(1-6), 66–92 (1960)
7. Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M.: Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata` (2007), release October, 12th 2007
8. Courcelle, B.: The monadic second-order logic of graphs. i. recognizable sets of finite graphs. Information and computation 85(1), 12–75 (1990)
9. Figueira, D.: On XPath with transitive axes and data tests. In: Hull, R., Fan, W. (eds.) PODS. pp. 249–260. ACM (2013)
10. Läuchli, H., Savioz, C.: Monadic second order definable relations on the binary tree. Journal of Symbol Logic 52(1), 219–226 (Mar 1987)
11. Müller, M., Niehren, J., Treinen, R.: The First-Order theory of ordering constraints over feature trees. In: LICS. pp. 432–443. IEEE Comp. Soc. Press (Jun 1998)
12. Niehren, J., Podelski, A.: Feature automata and recognizable sets of feature trees. In: TAPSOFT. LNCS, vol. 668, pp. 356–375. Springer (1993)
13. Rabin, M.: Automata on Infinite Objects and Church's Problem. No. 13 in CBMS Regional Conference Series in Mathematics, American Mathematical Society (1972)
14. Seidl, H., Schwentick, T., Muscholl, A.: Numerical document queries. In: Proceedings of the Symposium on Principles Of Database Systems. pp. 155–166 (2003)
15. Seidl, H., Schwentick, T., Muscholl, A.: Counting in trees. In: Logic and Automata. Texts in Logic and Games, vol. 2, pp. 575–612. Amsterdam University Press (2008)
16. Smolka, G.: Feature constraint logics for unification grammars. Journal of Logic Programming 12, 51–87 (1992)
17. Smolka, G., Treinen, R.: Records for logic programming. J. Log. Program. 18(3), 229–258 (1994)
18. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time. In: Symposium on the Theory of Computing. pp. 1–9. ACM (1973)
19. Thatcher, J.W., Wright, J.B.: Generalized finite automata theory with an application to a decision problem of second-order logic. MST 2(1), 57–81 (1968)
20. Zilio, S.D., Lugiez, D.: XML schema, tree logic and sheaves automata. In: Proc. of RTA. LNCS, vol. 2706, pp. 246–263. Springer Verlag (2003)