Alternation-free modal mu-calculus for data trees (Extended abstract)

Marcin Jurdziński and Ranko Lazić*
Department of Computer Science, University of Warwick, UK

Abstract

An alternation-free modal μ-calculus over data trees is introduced and studied. A data tree is an unranked ordered tree whose every node is labelled by a letter from a finite alphabet and an element ("datum") from an infinite set. For expressing data-sensitive properties, the calculus is equipped with freeze quantification. A freeze quantifier stores in a register the datum labelling the current tree node, which can then be accessed for equality comparisons deeper in the formula.

The main results in the paper are that, for the fragment with forward modal operators and one register, satisfiability over finite data trees is decidable but not primitive recursive, and that for the subfragment consisting of safety formulae, satisfiability over countable data trees is decidable but not elementary. The proofs use alternating tree automata which have registers, and establish correspondences with nondeterministic tree automata which have faulty counters. Allowing backward modal operators or two registers causes undecidability.

As consequences, decidability is obtained for two data-sensitive fragments of the XPath query language.

The paper shows that, for reasoning about data trees, the forward fragment of the calculus with one register is a powerful alternative to a recently proposed first-order logic with two variables.

1. Introduction

Context. Logics and automata for words and trees over finite alphabets are relatively well-understood. Motivated partly by the search for automated reasoning techniques for XML, and the need for formal verification and synthesis of infinite-state systems, there is an active and broad research programme on logics and automata for words and trees which have richer structure.

*Supported by a grant from the Intel Corporation.

The recent survey by Segoufin [30] summarises the substantial progress made on reasoning about data words and data trees. A data word is a word over a finite alphabet, with an equivalence relation on word positions. Implicitly, every word position is labelled by an element ("datum") from an infinite set ("data domain"), but since the infinite set is equipped only with the equality predicate, it suffices to know which word positions are labelled by equal data, and that is what the equivalence relation represents. Similarly, a data tree is a tree (countable, unranked and ordered) whose every node is labelled by a letter from a finite alphabet, with an equivalence relation on the set of its nodes.

First-order logic for data words was considered in [5], where variables range over word positions ($\{0, \ldots, l-1\}$ or $\mathbb{N}$), there is a unary predicate for each letter from the finite alphabet, and there is a binary predicate $x \sim y$ for the equivalence relation representing equality of data labels. $\mathrm{FO}^2(+1, <, \sim)$ denotes such a logic with two variables and binary predicates $x + 1 = y$ and $x < y$. Over finite and over infinite data words, satisfiability for $\mathrm{FO}^2(+1, <, \sim)$ was shown decidable and at least as hard as nonemptiness of vector addition automata [5]. Whether the latter problem is elementary has been open for many years. Extending the logic by one more variable causes undecidability [5].

Over data trees, $\mathrm{FO}^2(+1, <, \sim)$ denotes a similar first-order logic with two variables. The variables range over tree nodes, $+1$ stands for two predicates "child" and "next sibling", and $<$ stands for two predicates "descendant" and "younger sibling". In [4], complexity of satisfiability over finite data trees was studied. For $\mathrm{FO}^2(+1, \sim)$, it was shown to be in 3NEXPTIME, but for $\mathrm{FO}^2(+1, <, \sim)$, to be at least as hard as nonemptiness of vector addition tree automata. Decidability of the latter is a difficult open question, and it is equivalent to decidability of multiplicative exponential linear logic [10]. In [3], $\mathrm{FO}^2(+1, <, \sim)$ is shown decidable over finite data trees of fixed bounded depth.

XPath [7] is a prominent query language for XML documents [6]. The most basic static analysis problem for XPath, with a variety of applications, is satisfiability in the presence of DTDs. The two recent surveys on its complexity by Benedikt, Fan and Geerts [2, 17] convey that

22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)
0-7695-2908-9/07 $25.00 © 2007 IEEE

relatively little is known for fragments with negation and data (i.e., equality comparisons between attribute values). The only decidability result in [2, 17] that allows negation and data does not allow axes which are recursive (such as "self or descendant") or between siblings. By representing XML documents as data trees and translating from XPath to $\mathrm{FO}^2(+1, \sim)$, a decidable fragment with negation, data and all nonrecursive axes was obtained in [4]. Another fragment of XPath was considered in [18], but it lacks concatenation, recursive axes and sibling axes. Hence, decidability for the largest downward fragment, which was one of the main questions posed in [2], remains open.

An alternative approach to reasoning about data words was considered in [16, 23, 12, 11, 22], where expressiveness and algorithmic properties of linear temporal logic extended by freeze quantification (for short, $\mathrm{LTL}^{\downarrow}$) were studied. Freeze quantification was introduced in the context of timed logics (cf., e.g., [1]). A freeze quantifier $\downarrow_r$ stores in register r the equivalence class of the current word position. In its scope, the atomic formula $\uparrow_r$ is true at a word position iff the latter belongs to the equivalence class stored in r. Thus, freeze quantification enables data at different word positions to be compared for equality.

Over finite data words, satisfiability for $\mathrm{LTL}^{\downarrow}$ with future temporal operators and one register is decidable and not primitive recursive[1] [11]. Moreover, the problem becomes undecidable if infinite (i.e., ω) data words are considered, or if past temporal operators are allowed, or if one more register is available [11]. To overcome undecidability over infinite data words, the safety fragment of $\mathrm{LTL}^{\downarrow}$ with future temporal operators and one register was proposed in [22]. Whenever an infinite data word does not satisfy a safety sentence, it has a finite prefix whose every infinite extension also does not satisfy the sentence. The safety restriction does not affect expressiveness over finite data words. Over infinite data words, satisfiability for the safety fragment is EXPSPACE-complete, and refinement and model checking are decidable and not primitive recursive [22]. The fragment is not closed under negation, but decidability of refinement implies that satisfiability of Boolean combinations of safety sentences is decidable.

Contributions. This paper addresses one of the open research directions proposed in [30]: investigating computation tree logic extended by freeze quantification for reasoning about data trees. The principal logic we study, the alternation-free modal μ-calculus with freeze quantification, is more general. We focus on expressiveness and on complexity of satisfiability. Motivated partly by applications to XML data streams (cf., e.g., [27]), we consider both finite and countably infinite data trees. By finitary satisfiability, we mean satisfiability over finite data trees.

The main results are that, for the fragment with forward (i.e., downward and rightward) modal operators and one register, finitary satisfiability is decidable and not primitive recursive, and that for the safety subfragment, satisfiability is decidable and not elementary. We also show that satisfiability of Boolean combinations of sentences in the safety subfragment is decidable.

For the decidability results, the first step is translating from the logic to alternating tree register automata. Nondeterministic register automata were introduced in [19] (over data words) and [20] (over data trees). In contrast to the classical setting of the modal μ-calculus and finite automata over trees, alternation is needed for translating formulae with freeze quantification. In fact, already over data words, strict inclusions between deterministic and nondeterministic register automata, and between nondeterministic and alternating, were established in [19] and [25].

The second step consists of translations from alternating tree register automata to faulty counter automata over trees without data. The translations preserve nonemptiness, and involve nondeterminisation and quotienting. The counter automata are faulty in the sense that counters may spontaneously increase at any time. That feature makes their transition relations downwards compatible with a well-quasi-ordering (cf. [15]), upon which the proofs of decidability for those automata are based.

That satisfiability for the safety subfragment is not elementary is the most surprising result in the paper. It is relatively easy to obtain 2EXPTIME-hardness, by generalising the reduction that shows EXPSPACE-hardness for safety future $\mathrm{LTL}^{\downarrow}$ with one register [22]. It is then natural to expect that 2EXPTIME-membership for the safety subfragment over data trees can also be shown by extending the argument in [22] for EXPSPACE-membership.

By encoding computations of faulty counter tree automata as data trees, we provide a translation from the former to the forward alternation-free modal μ-calculus with one register, which closes the circle that was begun by the translations used for the decidability of finitary satisfiability. That yields a correspondence between languages of data trees defined by the forward fragment with one register and languages of trees defined by the counter automata.

It is worth noting that, with only forward modal operators and over finite data trees, there is no difference between least and greatest fixed points. Hence, the full modal mu-calculus, the alternation-free fragment and the safety subfragment all coincide in that setting. However, over arbitrary (i.e., possibly infinite) data trees, restricting to the alternation-free fragment enables us still to obtain a circle of translations as above and the consequent correspondence with languages of trees defined by faulty counter automata. That fact, and an equivalence with alternating tree register

[1] Recall the Ritchie-Cobham property [26, page 297]: a decision problem is primitive recursive iff it is solvable in primitive recursive time/space.

22nd Annual IEEE Symposium on Logic in Computer Science (LICS 2007)
0-7695-2908-9/07 $25.00 © 2007 IEEE

automata whose acceptance mechanism is weak parity, has led us to emphasise alternation freeness in this paper.

We show that $\mathrm{FO}^2(+1,\sim)$, the first-order logic for which finitary satisfiability was proved decidable in [4], is as expressive as a certain fragment of the alternation-free modal μ-calculus with one register, which is incomparable with the forward fragment. Indeed, forward modal operators can be iterated by means of fixed points, whereas the "descendant" and "younger sibling" predicates are not available in $\mathrm{FO}^2(+1,\sim)$. E.g., the "nonces" property (cf. Example 1) does not seem expressible in $\mathrm{FO}^2(+1,\sim)$. Moreover, temporal operators based on the "until" linear temporal operator are expressible in the forward fragment, but not even in $\mathrm{FO}^2(+1,<,\sim)$ (cf. [14]). On the other hand, the forward fragment has only forward modalities, whereas first-order quantifiers in $\mathrm{FO}^2(+1,\sim)$ range over all tree nodes.

We define forward XPath to be the largest downward and rightward fragment in which, whenever two attribute values are compared for equality, one of them must be at the current node. By translating from forward XPath to the forward alternation-free modal μ-calculus with one register, we obtain decidability of finitary satisfiability and decidability for a safety subfragment, both in the presence of DTDs. In contrast to the decidable fragments of XPath mentioned above, forward XPath has sibling axes, recursive axes, concatenation, negation, and data comparisons.

2. Preliminaries

Let $\mathbb{N}$ and $\mathbb{N}^+$ denote the sets of all nonnegative integers and of all positive integers, respectively. We write $\mathcal{P}(X)$ for the powerset of X.

For a relation R on a set X, we say that x is an R *predecessor* of y, or equivalently that y is an R *successor* of x, or equivalently that there is an R *edge* from x to y, iff $x\,R\,y$.

We say that a mapping f on a complete lattice $\langle X, \sqsubseteq\rangle$ is *upwards continuous* iff, for each chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots$ of elements of X, we have $f(\bigsqcup_{i\in\mathbb{N}} x_i) = \bigsqcup_{i\in\mathbb{N}} f(x_i)$. The least fixed point of such f is $\mathrm{lfp}(f) = \bigsqcup_{i\in\mathbb{N}} f^i(\bot)$, where $\bot$ is the least element of X. The notions of downward continuity and greatest fixed point are obtained by reversing the ordering. The latter is denoted by $\mathrm{gfp}(f)$.

2.1. Trees, contractions and data trees

Trees. We shall work with countable trees which are unranked, ordered, and whose every node is labelled by a letter from a finite alphabet. More precisely, a *tree* will be a tuple $\langle N, \Sigma, n_R, \triangledown, \triangleright, \lambda\rangle$ such that: N is a countable set of nodes; Σ is a finite alphabet; $n_R \in N$ is the root node; $\triangledown$ is the "child" relation on N; $\triangleright$ is the "next sibling" relation on N; each node has an oldest sibling, from which it is reachable by a finite number of $\triangleright$ edges; n_R is an oldest sibling, and each node is reachable from n_R or a sibling of n_R by a finite number of $\triangledown$ edges; $\lambda : N \to \Sigma$ is a labelling.

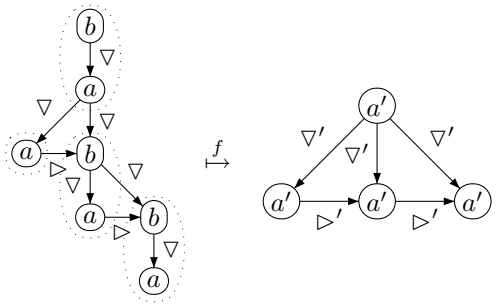

Figure 1. Applying $f = \{a \mapsto a', b \mapsto \varepsilon\}$

Let $\triangle$ and $\triangleleft$ denote the inverses of $\triangledown$ and $\triangleright$, and $\triangledown_1$ be the "first child" relation, i.e., $n \triangledown_1 n'$ iff $n \triangledown n'$ and $n' \not\triangleleft$.

As can be seen in the definition above, the trees we consider in this paper are actually forests, where "the root" is the oldest node with no parent. Like in [3], we take that approach for technical reasons. However, each $\langle N, \Sigma, n_R, \triangledown, \triangleright, \lambda\rangle$ as above can be seen as a binary tree with root n_R, where left-hand and right-hand children are given by relations $\triangledown_1$ and $\triangleright$, respectively. In fact, those two relations will be used for navigation in the forward modal μ-calculus, forward register automata, and counter automata (cf. Sections 2.2, 2.3 and 2.4). Hence, we shall refer to the defined structures as trees.

By König's Lemma, a tree is infinite iff some node has infinitely many siblings or some downward path is infinite.

Contractions. We now define a class of operations on trees, which will be used in Section 2.4 for extracting trees from accepting runs of automata with ε-transitions.

A *contraction* is a mapping $f : \Sigma \to \Sigma' \uplus \{\varepsilon\}$, where Σ and Σ' are finite alphabets and ε is the empty word. Given a tree t as above, we say that $n \in N$ is an *ε-node* iff $f(\lambda(n)) = \varepsilon$. We say that t is *f-good* iff: no ε-node has a next sibling, and from every ε-node, some non-ε node is reachable by $\triangledown$ edges. For such f and t, the tree $f(t) = \langle N', \Sigma', n'_R, \triangledown', \triangleright', \lambda'\rangle$ is defined as follows. An example is shown in Figure 1.

- The nodes N' are maximal paths $n_1 \triangledown_1 \cdots \triangledown_1 n_k$ such that $n_1, \ldots, n_{k-1}$ are ε-nodes and n_k is not.
- The root n'_R is the path which contains n_R.
- $(n_1 \triangledown_1 \cdots \triangledown_1 n_k) \triangledown' (m_1 \triangledown_1 \cdots \triangledown_1 m_k)$ iff $n_k \triangledown m_1$.
- $(n_1 \triangledown_1 \cdots \triangledown_1 n_k) \triangleright' (m_1 \triangledown_1 \cdots \triangledown_1 m_k)$ iff $n_k \triangleright m_1$.
- $\lambda'(n_1 \triangledown_1 \cdots \triangledown_1 n_k) = f(\lambda(n_k))$.

Data trees. We shall mainly consider trees whose every node is also labelled by an element from an infinite set

which is equipped only with equality. Formally, a *data tree* is a tree as above, with an equivalence relation $\sim$ on N.

For $n \in N$, let $[n]_\sim$ denote the class of $\sim$ which contains n. We write $N/\sim$ for the set of all classes of $\sim$. We shall use the terms 'datum' and 'class of $\sim$' interchangably. For a data tree τ, let $\mathrm{tree}(\tau)$ be the underlying tree.

2.2. Alternation-free modal μ-calculus with freeze quantification

The central logic in this paper is based on the alternation-free modal μ-calculus [13]. For equiexpressiveness with automata (cf. Section 2.3) and succinctness (cf. Theorem 19), instead of having fixed-point operators, we work with systems of equations. Each equation has a type: if μ, the equation is to be solved by finding the least set of nodes which satisfies it; if ν, the greatest solution is sought. Alternation freeness ensures that there are no mutual equations whose types are different.

Each system is over a finite alphabet Σ, and it is satisfied at a node of a data tree with alphabet Σ iff the node belongs to the solution of its last equation. For each $a \in \Sigma$, atomic formula a is true iff the current node is labelled by a. There are four modal operators, for asserting that a subformula is true at the first child, parent, next sibling, or previous sibling node. For each of the four kinds of neighbour, we have an atomic proposition which is true iff that neighbour exists. Although those are expressible by the modal operators and $\top$, they are useful to have in fragments which contain only some of the modal operators. To express data-sensitive properties, we use the freeze quantifier $\downarrow_r$ which stores the datum labelling the current node in register $r \in \mathbb{N}^+$, and an atomic formula $\uparrow_r$ which is true iff the datum in register r equals the datum labelling the current node.

Syntax and fragments. The formulae of the modal μ-calculus with freeze quantification are defined by the following grammar, where V ranges over an infinite set of variables. We work with negation normal forms, so there is no $\neg$ operator. The atomic formulae $\overline{a}$, $\overline{\texttt{down}}$, $\overline{\texttt{up}}$, $\overline{\texttt{right}}$, $\overline{\texttt{left}}$ and $\not\uparrow_r$ represent negations of a, $\texttt{down}$, $\texttt{up}$, $\texttt{right}$, $\texttt{left}$ and $\uparrow_r$, respectively. In their scopes, the freeze quantifiers $\downarrow_r$ bind occurences of $\uparrow_r$ and $\not\uparrow_r$.

$$\begin{array}{rcl} \varphi & ::= & V \mid a \mid \overline{a} \mid \top \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \\ & & \texttt{down} \mid \texttt{up} \mid \texttt{right} \mid \texttt{left} \mid \\ & & \overline{\texttt{down}} \mid \overline{\texttt{up}} \mid \overline{\texttt{right}} \mid \overline{\texttt{left}} \mid \\ & & \Diamond\varphi \mid \Box\varphi \mid \Diamond^-\varphi \mid \Box^-\varphi \mid \downarrow_r\varphi \mid \uparrow_r \mid \not\uparrow_r \end{array}$$

A system σ of the full calculus is a nonempty sequence

$$V_1 \stackrel{\pi_1}{=} \varphi_1, \ldots, V_k \stackrel{\pi_k}{=} \varphi_k$$

where, for each i, V_i is a variable, π_i is a type (either μ or ν), and φ_i is a formula. We say that the system is *closed* iff it contains no variables other than $V_1, \ldots, V_k$.

We restrict our attention to alternation-free systems, which are defined as follows. For every $i \in \{1, \ldots, k\}$, we write $\mathrm{block}(i)$ for the maximum subset of $\{1, \ldots, k\}$ that contains i, that is contiguous, and for whose every member j we have $\pi_j = \pi_i$. A system as above is *alternation free* iff, whenever V_j occurs in φ_i, either $j \in \mathrm{block}(i)$ or $j < i$.

We say that a system is *guarded* iff, whenever V_j occurs in φ_i not under a modal operator (i.e., $\Diamond$, $\Box$, $\Diamond^-$ or $\Box^-$), we have $j < i$. We shall consider only guarded systems. That expressiveness is not affected by that constraint will follow from a remark before Theorem 3 and Theorem 7.

We write $\mu^\downarrow(\mathcal{M})$ for the fragment of the alternation-free modal μ-calculus with freeze quantification which has only modal operators from the set $\mathcal{M} \subseteq \{\Diamond, \Box, \Diamond^-, \Box^-\}$. We consider subfragments defined by restricting to a finite alphabet Σ and/or $k \in \mathbb{N}$ registers (i.e., with $r \in \{1, \ldots, k\}$). For example, $\mu^\downarrow(\Diamond, \Diamond^-)$ is the *forward* fragment, and $\mu^{\downarrow,1}_\Sigma(\Diamond, \Diamond^-)$ is the subfragment with alphabet Σ and one register.

In a *safety* (resp., *co-safety*) system, all fixed points are greatest (resp., least), i.e., each type is ν (resp., μ).

Semantics. For a data tree $\tau = \langle N, \Sigma, n_R, \nabla, \triangleright, \lambda, \sim \rangle$, a *variable valuation* is a mapping from a finite subset of the set of all variables to $\mathcal{P}(N)$, and a *register valuation* is a mapping from a finite subset of $\mathbb{N}^+$ to $N/\sim$.

If φ is a formula over Σ, u is a variable valuation for τ which is defined for every variable occuring in φ, v is a register valuation for τ, and $n \in N$, we write $\tau, n \models_{u,v} \varphi$ iff τ at n satisfies φ with respect to u and v. The definition of that relation is recursive over φ and standard, so we omit the Boolean and dual cases:

- $\tau, n \models_{u,v} V$ iff $n \in u(V)$;
- $\tau, n \models_{u,v} a$ iff $\lambda(n) = a$;
- $\tau, n \models_{u,v} \{\texttt{down}, \texttt{up}, \texttt{right}, \texttt{left}\}$ iff there exists n' such that $n\{\nabla_1, \vartriangle, \triangleright, \triangleleft\}n'$ (respectively);
- $\tau, n \models_{u,v} \{\Diamond, \Box, \Diamond^-, \Box^-\}\varphi$ iff there exists n' such that $n\{\nabla_1, \vartriangle, \triangleright, \triangleleft\}n'$ (respectively) and $\tau, n' \models_{u,v} \varphi$;
- $\tau, n \models_{u,v} \downarrow_r\varphi$ iff $\tau, n \models_{u,v[r \mapsto [n]_\sim]} \varphi$;
- $\tau, n \models_{u,v} \uparrow_r$ iff $r \in \mathrm{dom}(v)$ and $n \in v(r)$.

If σ is an alternation-free system $V_1 \stackrel{\pi_1}{=} \varphi_1, \ldots, V_k \stackrel{\pi_k}{=} \varphi_k$ over Σ, u is a variable valuation for τ which is defined for every variable occuring free in σ, and v is a register valuation for τ, we write $[\![\sigma]\!]^\tau_{u,v}$ to denote the valuation of $\{V_1, \ldots, V_k\}$ for τ that is determined by the equations with respect to u and v. The definition is recursive over the number of blocks in σ:

- In the base case, there is only one block. Let F be the following mapping on $\mathcal{P}(N)^k$ which is upwards and downwards continuous with respect to the pointwise

subset ordering:

$$F\langle M_1,\ldots,M_k\rangle = \langle\{m \in N : \tau, m \models_{u',v} \varphi_1\}, \ldots, \{m \in N : \tau, m \models_{u',v} \varphi_k\}\rangle$$

where $u' = u[V_1 \mapsto M_1, \ldots, V_k \mapsto M_k]$. If $\pi_1, \ldots, \pi_k$ are μ, then $[\![\sigma]\!]^\tau_{u,v}$ is defined as $\mathrm{lfp}(F)$. Otherwise, $\pi_{k'+1}, \ldots, \pi_k$ are ν, and $[\![\sigma]\!]^\tau_{u,v}$ is defined as $\mathrm{gfp}(F)$.

- When there is more than one block, let k' be the maximum in $\{1,\ldots,k\} \setminus \mathrm{block}(k)$. To solve σ, the first k' equations are solved recursively, and then the equations in the last block are solved as in the base case:

$$[\![\sigma]\!]^\tau_{u,v} = u' \uplus [\![V_{k'} \stackrel{\pi_{k'}}{=} \varphi_{k'}, \ldots, V_k \stackrel{\pi_k}{=} \varphi_k]\!]^\tau_{u \uplus u',v}$$

where $u' = [\![V_1 \stackrel{\pi_1}{=} \varphi_1, \ldots, V_{k'} \stackrel{\pi_{k'}}{=} \varphi_{k'}]\!]^\tau_{u,v}$.

For closed σ, its language $\mathrm{L}_\Sigma(\sigma)$ is defined as the set of all data trees τ with alphabet Σ such that $n_R \in [\![\sigma]\!]^\tau_{\emptyset,\emptyset}(V_k)$. We write $\mathrm{L}^{<\omega}_\Sigma(\sigma)$ for the subset of all finite data trees in $\mathrm{L}_\Sigma(\sigma)$. We say that σ is satisfiable (resp., finitely satisfiable) iff $\mathrm{L}_\Sigma(\sigma)$ (resp., $\mathrm{L}^{<\omega}_\Sigma(\sigma)$) is nonempty.

Boolean operations. For closed σ, its dual (i.e., negation) $\overline{\sigma}$ is defined as $V_1 \stackrel{\overline{\pi_1}}{=} \overline{\varphi_1}, \ldots, V_k \stackrel{\overline{\pi_k}}{=} \overline{\varphi_k}$, where for each i, $\overline{\pi_i} = \mu$ iff $\pi_i = \nu$ and $\overline{\varphi_i}$ is obtained from φ_i by replacing each construct with its dual. The duals of $\circledcirc\varphi$, $\circledast\varphi$, $\circledominus\varphi$ and $\circledoteq\varphi$ are $\overline{\mathtt{down}} \vee \circledcirc\overline{\varphi}$, $\overline{\mathtt{up}} \vee \circledast\overline{\varphi}$, $\overline{\mathtt{right}} \vee \circledominus\overline{\varphi}$ and $\overline{\mathtt{left}} \vee \circledoteq\overline{\varphi}$, respectively. The freeze quantifiers $\downarrow_r$ are self-dual. By inductions over formulae and the number of blocks, we have that $[\![\overline{\sigma}]\!]^\tau_{\emptyset,v}(V_i) = N \setminus [\![\sigma]\!]^\tau_{\emptyset,v}(V_i)$ for each i.

The conjunction (resp., disjunction) of two systems $V_1 \stackrel{\pi_1}{=} \varphi_1, \ldots, V_k \stackrel{\pi_k}{=} \varphi_k$ and $V'_1 \stackrel{\pi'_1}{=} \varphi'_1, \ldots, V'_{k'} \stackrel{\pi'_{k'}}{=} \varphi'_{k'}$ over the same alphabet is defined as their concatenation followed by an equation $V'' \stackrel{\pi''}{=} \varphi''$, where π'' is arbitrary and φ'' is $V_k \wedge V'_{k'}$ (resp., $V_k \vee V'_{k'}$).

We say that a closed system σ is valid iff $\overline{\sigma}$ is not satisfiable. For closed systems σ and σ' over the same alphabet, we say that σ refines (resp., finitely refines) σ' iff $\sigma \Rightarrow \sigma'$ (i.e., $\overline{\sigma} \vee \sigma'$) is valid (resp., finitely valid).

Expressing CTL operators. We consider operators of computation tree logic [8] of the form $\mathtt{Q}\,\mathtt{L}$ for directed path quantifiers $\mathtt{Q} \in \{\triangledown\mathtt{E}, \triangledown\mathtt{A}, \triangle, \triangleright, \triangleleft\}$ and linear-time operators $\mathtt{L} \in \{\mathtt{X}, \mathtt{U}, \mathtt{R}\}$. The quantifiers $\triangledown\mathtt{E}$ and $\triangledown\mathtt{A}$ are existential and universal (respectively) over all maximal downward paths, where a path is downward iff it consists of $\triangledown$ edges. Unique maximal upward, rightward and leftward paths are specified by the quantifiers $\triangle$, $\triangleright$ and $\triangleleft$, respectively. The "release" operator $\mathtt{R}$ is the dual of the "until" operator $\mathtt{U}$. The "eventually" and "always" operators $\mathtt{F}\phi$ and $\mathtt{G}\phi$ can be defined as $\top\mathtt{U}\phi$ and $\bot\mathtt{R}\phi$, respectively.

We say that a CTL operator $\mathtt{Q}\,\mathtt{L}$ is existential iff $\mathtt{Q} \in \{\triangledown\mathtt{E}, \triangle, \triangleright, \triangleleft\}$ and $\mathtt{L} \in \{\mathtt{X}, \mathtt{U}\}$. Those operators can be expressed in $\mu^{\downarrow}(\circledcirc, \circledast, \circledominus, \circledoteq)$ as below. Their duals are the universal CTL operators, i.e., $\mathtt{Q}\,\mathtt{L}$ where $\mathtt{Q} \in \{\triangledown\mathtt{A}, \triangle, \triangleright, \triangleleft\}$ and $\mathtt{L} \in \{\mathtt{X}, \mathtt{R}\}$, so the latter can also be expressed. In the definitions, V and V' denote the last left-hand sides in systems σ and σ', and irrelevant types are omitted.

$$\begin{aligned}
\{\triangle,\triangleright,\triangleleft\}\mathtt{X}\sigma &\stackrel{\mathrm{def}}{=} \sigma, W = \{\circledast, \circledominus, \circledoteq\}V\\
\{\triangle,\triangleright,\triangleleft\}[\sigma\mathtt{U}\sigma'] &\stackrel{\mathrm{def}}{=} \sigma, \sigma',\\
&\quad W \stackrel{\mu}{=} V' \vee (V \wedge \{\circledast, \circledominus, \circledoteq\}W)\\
\triangledown\mathtt{EX}\sigma &\stackrel{\mathrm{def}}{=} \sigma, W \stackrel{\mu}{=} V \vee \circledominus W, W' = \circledcirc W\\
\triangledown\mathtt{E}[\sigma\mathtt{U}\sigma'] &\stackrel{\mathrm{def}}{=} \sigma, \sigma', W \stackrel{\mu}{=} V' \vee (V \wedge \circledcirc W'),\\
&\quad W' \stackrel{\mu}{=} W \vee \circledominus W', W'' = W
\end{aligned}$$

Example 1 With the definitions above, we have that the sentence $\triangleright\mathtt{G}\triangledown\mathtt{AG}(\downarrow_1 \triangledown\mathtt{AX}\triangledown\mathtt{AG}\neg\uparrow_1)$ is expressible as a safety closed system in $\mu^{\downarrow,1}(\circledcirc, \circledominus)$, the forward fragment with one register. It is satisfied by a data tree iff, for each node, there is no descendant which is labelled by the same datum. □

A consequence of Corollary 5 below will be that, when considering only finite data trees, the two remaining CTL operators $\triangledown\mathtt{ER}$ and $\triangledown\mathtt{AU}$ are also expressible.

Lower bounds for satisfiability. The following theorem contains a number of lower bounds for the complexity of deciding satisfiability for fragments of the alternation-free modal μ-calculus with freeze quantification. They are obtained from similar results in [16, 9, 11], which are for LTL with freeze quantification. Closely matching upper bounds will be established in Theorem 6 and Corollary 10 below.

Theorem 2 *(a) For $\mu^{\downarrow,1}(\circledcirc)$ and $\mu^{\downarrow,1}(\circledominus)$, fin. satisfiability is not primitive recursive and satisfiability is Π^0_1-hard. (b) For $\mu^{\downarrow,1}(\circledcirc, \circledast)$, $\mu^{\downarrow,1}(\circledominus, \circledoteq)$, $\mu^{\downarrow,2}(\circledcirc)$ and $\mu^{\downarrow,2}(\circledominus)$, fin. satisfiability is Σ^0_1-hard and satisfiability is Σ^1_1-hard.*

2.3. Alternating tree register automata

Corresponding to the addition of freeze quantification to the modal μ-calculus, finite automata can be extended by registers. We now define alternating register automata over data trees, and establish their equiexpressiveness with the calculus. For running over trees, the automata can move to the first child, parent, previous sibling or next sibling nodes. Existential and universal branchings are expressed by transition formulae which are disjunctions and conjunctions, respectively. Using atomic transition formulae, the automata can check whether a move down, up, right or left is possible. Consequently, there is no need for final locations. Each location has a rank, in terms of which weak parity acceptance will be defined in a standard manner [21]. Weak parity acceptance is self-dual, is a special case of Büchi acceptance, and suffices for translating alternation-free modal μ-calculus to alternating automata. Locations also have

heights, which ensure that the automata cannot make infinite progress without performing a move. That constraint, which corresponds to system guardedness, simplifies some proofs while not reducing expressiveness. It enables runs to be defined using "big-step" transitions, which are sequences of moveless "small-step" transitions followed by a move.

Automata. The set $\Phi(\Sigma, Q, k)$ of all transition formulae over a finite alphabet Σ, over a finite set Q of locations and with $k \in \mathbb{N}$ registers is defined as:

$$\{a, \overline{a}, \top, \bot, q \wedge q', q \vee q', \mathtt{down}, \mathtt{up}, \mathtt{right}, \mathtt{left}, \overline{\mathtt{down}}, \overline{\mathtt{up}}, \overline{\mathtt{right}}, \overline{\mathtt{left}}, \circledcirc q, \circledast q, \circledcirc q, \circledast q, \downarrow_r q, \uparrow_r, \not\downarrow_r : a \in \Sigma, q, q' \in Q, r \in \{1, \ldots, k\}\}$$

An *alternating tree register automaton* $\mathcal{A}$ is a tuple $\langle \Sigma, Q, q_I, k, \delta, \rho, \gamma \rangle$ such that:

- Σ is a finite alphabet;
- Q is a finite set of locations;
- $q_I \in Q$ is the initial location;
- $k \in \mathbb{N}$ is the number of registers;
- $\delta : Q \to \Phi(\Sigma, Q, k)$ is a transition function;
- $\rho : Q \to \mathbb{N}$ specifies ranks and is such that, whenever q' occurs in $\delta(q)$, we have $\rho(q') \leq \rho(q)$;
- $\gamma : Q \to \mathbb{N}$ specifies heights and is such that, whenever q' occurs in $\delta(q)$ which is not a move formula (i.e., not one of $\circledcirc q'$, $\circledcirc q'$, $\circledast q'$ or $\circledast q'$), we have $\gamma(q') < \gamma(q)$.

We write $\mathrm{ATRA}(\mathcal{M})$ for the set of all automata which contain only moves from the set $\mathcal{M} \subseteq \{\circledcirc, \circledcirc, \circledast, \circledast\}$. We consider subsets defined by restricting to automata with a specified alphabet and/or a specified number of registers. For example, $\mathrm{ATRA}^1(\circledcirc, \circledast)$ is the set of all *forward* automata with one register.

A *safety* (resp., *co-safety*) automaton is one in which each location rank is even (resp., odd).

Runs and languages. Let $\tau = \langle N, \Sigma, n_R, \nabla, \triangleright, \lambda, \sim \rangle$ be a data tree. To define runs of $\mathcal{A}$ over τ, we first define a state of $\mathcal{A}$ for τ to be a triple $\langle n, q, v \rangle$ where $n \in N$, $q \in Q$, and v is a register valuation for τ.

For $\phi \in \Phi(\Sigma, Q, k)$, let $[\![\phi]\!]_v^{\tau,n}$ denote the set of sets of states that results from ϕ with respect to τ, n and v. The following are the representative cases:

$$\begin{aligned}
[\![q \wedge q']\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \{\{\langle n, q, v\rangle, \langle n, q', v\rangle\}\} \\
[\![q \vee q']\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \{\{\langle n, q, v\rangle\}, \{\langle n, q', v\rangle\}\} \\
[\![\mathtt{down}]\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \begin{cases} \{\emptyset\}, & \text{if } n \nabla_1 n' \text{ for some } n' \\ \emptyset, & \text{otherwise} \end{cases} \\
[\![\circledcirc q]\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \begin{cases} \{\{\langle n', q, v\rangle\}\}, & \text{if } n \nabla_1 n' \\ \emptyset, & \text{otherwise} \end{cases} \\
[\![\downarrow_r q]\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \{\{\langle n, q, v[r \mapsto [n]_\sim]\rangle\}\} \\
[\![\uparrow_r]\!]_v^{\tau,n} &\stackrel{\text{def}}{=} \begin{cases} \{\emptyset\}, & \text{if } r \in \mathrm{dom}(v), n \in v(r) \\ \emptyset, & \text{otherwise} \end{cases}
\end{aligned}$$

We write $\langle n, q, v \rangle \rightsquigarrow S$ iff S is a set of states which may result by performing transitions from $\langle n, q, v \rangle$ up to first moves. The relation is defined recursively over $\gamma(q)$: if $\delta(q)$ is a move formula, then $\langle n, q, v \rangle \rightsquigarrow S$ iff $S \in [\![\delta(q)]\!]_v^{\tau,n}$; otherwise, $\langle n, q, v \rangle \rightsquigarrow S$ iff $S = \bigcup_{\langle n', q', v' \rangle \in S'} S''_{\langle n', q', v' \rangle}$ for some $S' \in [\![\delta(q)]\!]_v^{\tau,n}$ and, for each $\langle n', q', v' \rangle \in S'$, some $S''_{\langle n', q', v' \rangle}$ such that $\langle n', q', v' \rangle \rightsquigarrow S''_{\langle n', q', v' \rangle}$. Observe that, for every $\langle n, q, v \rangle$, there are finitely many S with $\langle n, q, v \rangle \rightsquigarrow S$, and each such S is finite.

Now, a run G of $\mathcal{A}$ of length $0 < \kappa \leq \omega$ over τ is a sequence $\langle G(i) : 0 \leq i < \kappa \rangle$ of sets of states of $\mathcal{A}$ for τ, together with relations $\to_i \subseteq G(i) \times G(i+1)$, such that:

- $G(0) = \{\langle n_R, q_I, \emptyset \rangle\}$;
- for every $i + 1 < \kappa$, $G(i+1) = \bigcup_{\langle n, q, v \rangle \in G(i)} S^i_{\langle n, q, v \rangle}$ for some $\langle n, q, v \rangle \rightsquigarrow S^i_{\langle n, q, v \rangle}$;
- $\langle n, q, v \rangle \to_i \langle n', q', v' \rangle$ iff $\langle n', q', v' \rangle \in S^i_{\langle n, q, v \rangle}$.

A run G of length $\kappa < \omega$ is considered accepting iff $G(\kappa - 1) = \emptyset$. Suppose G is an infinite run, and θ is an infinite thread $\langle n_R, q_I, \emptyset \rangle = \langle n_0, q_0, v_0 \rangle \to_0 \langle n_1, q_1, v_1 \rangle \to_1 \cdots$ in G. We have $\rho(q_0) \geq \rho(q_1) \geq \cdots$, so we can define $\rho(\theta)$ as the eventually constant location rank. We consider G accepting iff, for each infinite thread θ, $\rho(\theta)$ is even.

We say that $\mathcal{A}$ accepts τ iff $\mathcal{A}$ has an accepting run over τ. The language $\mathrm{L}(\mathcal{A})$ is the set of all data trees τ with alphabet Σ which are accepted by $\mathcal{A}$. We write $\mathrm{L}^{<\omega}(\mathcal{A})$ for the subset of all finite data trees in $\mathrm{L}(\mathcal{A})$. We say that $\mathcal{A}$ is nonempty (resp., finitely nonempty) iff $\mathrm{L}(\mathcal{A})$ (resp., $\mathrm{L}^{<\omega}(\mathcal{A})$) is nonempty.

From logic to automata... The following result is the first step towards several upper complexity bounds for satisfiability for fragments of the calculus, which will be established in Theorem 6, Corollary 10 and Theorem 15. We remark that unguarded systems can be translated in logarithmic space to heightless automata, but the latter are translatable to heighted automata in polynomial space.

Theorem 3 *For each closed system σ of $\mu^{\downarrow,k}_\Sigma(\mathcal{M})$, an automaton $\mathcal{A}^\Sigma_\sigma$ in $\mathrm{ATRA}^k_\Sigma(\mathcal{M})$, such that $\mathrm{L}_\Sigma(\sigma) = \mathrm{L}(\mathcal{A}^\Sigma_\sigma)$, is computable in logarithmic space. If σ is safety (resp., co-safety), then so is $\mathcal{A}^\Sigma_\sigma$.*

Example 4 Let $\triangleright \mathtt{G} \nabla \mathtt{AG}(\downarrow_1 \nabla \mathtt{AX} \nabla \mathtt{AG} \neg \uparrow_1)$ be the "nonces" sentence from Example 1. The automaton pictured in Figure 2 has the same language, is in $\mathrm{ATRA}^1(\circledcirc, \circledast)$, and is safety. For readability, we used ternary conjunctions, and $\overline{\circledcirc} q$ and $\overline{\circledast} q$ to abbreviate $\overline{\mathtt{down}} \vee \circledcirc q$ and $\overline{\mathtt{right}} \vee \circledast q$ (respectively). Each location has even rank (e.g., 0), and suitable heights are straightforward to assign. □

By König's Lemma, every infinite run has an infinite thread. Therefore, if $\mathcal{A}$ is a forward automaton and τ is

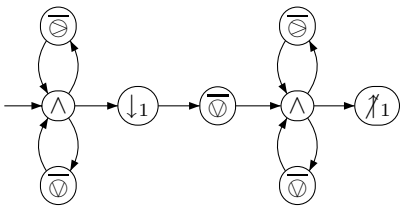

Figure 2. An alternating tree reg. automaton

finite, then every run of $\mathcal{A}$ over τ is finite. From that observation and the translation in the proof of Theorem 3, we obtain the next corollary. It shows that, with only forward modal operators and over finite data trees, safety is not a restriction (and neither is alternation freeness).

Corollary 5 *If closed systems σ and σ' of $\mu^{\downarrow}_{\Sigma}(\Diamond, \Diamond)$ differ only in types, then $\mathrm{L}^{<\omega}_{\Sigma}(\sigma) = \mathrm{L}^{<\omega}_{\Sigma}(\sigma')$.*

Using Theorem 3 and the concept of consistent signature assignment [31], the following two basic upper complexity bounds for satisfiability can be obtained. They match the lower bounds in Theorem 2(b).

Theorem 6 *For $\mu^{\downarrow}(\Diamond, \Box, \Diamond, \Box)$, finitary satisfiability is in Σ^0_1 and satisfiability is in Σ^1_1.*

...and back. Together with Theorem 3, the next result shows that the alternation-free modal μ-calculus with freeze quantification is as expressive as alternating tree register automata, and without significant differences in succinctness.

Theorem 7 *For each $\mathcal{A}$ in $\mathrm{ATRA}^k_{\Sigma}(\mathcal{M})$, a closed system $\sigma_{\mathcal{A}}$ of $\mu^{\downarrow,k}_{\Sigma}(\mathcal{M})$, such that $\mathrm{L}(\mathcal{A}) = \mathrm{L}_{\Sigma}(\sigma_{\mathcal{A}})$, is computable in logarithmic space. Safety and co-safety are preserved.*

2.4. Faulty tree counter automata

In Section 3, we shall establish a correspondence between closed systems of the forward alternation-free modal μ-calculus with one register and faulty tree automata with counters. We now define the latter, and determine the complexity of deciding their nonemptiness.

Each automaton will have a finite number of natural-valued counters, on which increment, decrement (if nonzero) and zero-test instructions are available. It will run over trees in a top-down nondeterministic manner: if the current node is labelled by a letter a, an a-transition can be performed, producing states for the first child and next sibling nodes (if any); alternatively, performing an ε-transition can change the state, but we remain at the same node. The automata will be faulty in the sense that runs can contain errors which increase one or more counters. For acceptance, there will be two "lights": "final" and "Büchi". A run will be accepting iff each location produced for the first child of a leaf node is final, each location produced for the next sibling of a last sibling is final, and each infinite sequence of transitions contains a Büchi location infinitely often. The formalisation below, where we omit technical details, is suited to the uses in Section 3.

Automata. An *incrementing tree counter automaton* (ITCA) $\mathcal{C}$, which is top-down, with ε-transitions, and with zero testing, is a tuple $\langle \Sigma, Q, q_I, k, \delta, F, B \rangle$ such that: Σ is a finite alphabet; Q is a finite set of locations; q_I is the initial location; $k \in \mathbb{N}$ is the number of counters; $\delta \subseteq (Q \times \Sigma \times L \times Q \times Q) \cup (Q \times \{\varepsilon\} \times L \times Q)$ is a transition relation, where $L = \{\mathtt{inc}, \mathtt{dec}, \mathtt{ifzero}\} \times \{1, \ldots, k\}$ is the instruction set; $F, B \subseteq Q$ are the sets of final and Büchi locations (respectively), such that whenever $\langle q, \varepsilon, l, q' \rangle \in \delta$, we have $q' \notin F, B$.

Runs and languages. A counter valuation is a function $\{1, \ldots, k\} \to \mathbb{N}$. Let $\Sigma^{\dagger} = (Q \times \Sigma \times L \times Q \times Q) \cup (Q \times \{\varepsilon\} \times L \times Q)$. A run of $\mathcal{C}$ is a tree $t = \langle N, \Sigma^{\dagger}, n_R, \triangledown, \triangleright, \lambda \rangle$ together with mappings β and β' from N to counter valuations. For each $n \in N$, $\beta(n)$ is the counter valuation before performing the transition $\lambda(n)$, and $\beta'(n)$ is a counter valuation after the transition is performed.

Let $\mathrm{proj} : \Sigma^{\dagger} \to \Sigma \uplus \{\varepsilon\}$ be the contraction defined by $\mathrm{proj}(\langle q, a, l, q', q'' \rangle) = a$ and $\mathrm{proj}(\langle q, \varepsilon, l, q' \rangle) = \varepsilon$. For every accepting run $\langle t, \beta, \beta' \rangle$, we have that t is proj-good (cf. Section 2.1). We say that $\mathcal{C}$ accepts a tree t' with alphabet Σ iff $\mathcal{C}$ has an accepting run $\langle t, \beta, \beta' \rangle$ such that $t' = \mathrm{proj}(t)$. The language $\mathrm{L}(\mathcal{C})$, nonemptiness of $\mathcal{C}$, and their finitary versions, are defined as usual.

Complexity of nonemptiness. As is well-known, without incrementing errors and already over words, finitary nonemptiness of deterministic 2-counter automata is undecidable (more precisely, Σ^0_1-hard) [24], and nonemptiness of 2-counter automata is Σ^1_1-hard [1, Lemma 8]. In the theorem below, decidability and Π^0_1-membership are shown using well-quasi-orderings (cf., e.g., [15]). Non-primitive recursiveness and Π^0_1-hardness follow from results in [29, 28].

Theorem 8 *For incrementing tree counter automata, finitary nonemptiness is decidable and not primitive recursive, and nonemptiness is Π^0_1-complete.*

3. Correspondence with counter automata

The next theorem states that, given a forward alternating tree automaton $\mathcal{A}$ with one register, an incrementing tree counter automaton $\mathcal{C}_{\mathcal{A}}$, which is nonempty iff $\mathcal{A}$ is nonempty, is computable in polynomial space. It is proved by extending to trees the translation in the proof of [11,

Theorem 12], which is from one-way alternating word automata with one register to incrementing word counter automata. The theorem enables us to conclude, using Theorems 3 and 8 above, that for the forward alternation-free modal μ-calculus with one register, finitary satisfiability is decidable, and satisfiability is co-r.e.

Theorem 9 *For each $\mathcal{A}$ in ATRA$^1(\Diamond, \Diamond)$, an ITCA $\mathcal{C}_\mathcal{A}$, with the same alphabet and such that $\mathrm{L}(\mathcal{C}_\mathcal{A}) = \{\mathrm{tree}(\tau) : \tau \in \mathrm{L}(\mathcal{A})\}$, is computable in polynomial space.*

Corollary 10 *For $\mu^{\downarrow,1}(\Diamond, \Diamond)$, finitary satisfiability is decidable and satisfiability is in Π^0_1.*

The following result completes the correspondence between the forward alternation-free modal μ-calculus with one register and incrementing tree counter automata. It shows that, for each ITCA $\mathcal{C}$, a closed system whose models are exactly all accepting runs of $\mathcal{C}$ is computable in logarithmic space. We then recall Theorems 3 and 9 above to infer that contractions of projections of languages of closed systems in the fragment coincide with languages of ITCA.

Theorem 11 *For each ITCA $\mathcal{C}$ with alphabet Σ, a closed system $\sigma_\mathcal{C}$ of $\mu^{\downarrow,1}_{\Sigma^\dagger}(\Diamond, \Diamond)$, such that the set of all accepting runs of $\mathcal{C}$ equals $\{\mathrm{tree}(\tau) : \tau \in \mathrm{L}_{\Sigma^\dagger}(\sigma_\mathcal{C})\}$, is computable in logarithmic space.*

Corollary 12 *For every finite alphabet Σ', the set of all $\mathrm{L}(\mathcal{C})$ for ITCA $\mathcal{C}$ with alphabet Σ' is equal to the set of all $\{f(\mathrm{tree}(\tau)) : \tau \in \mathrm{L}_\Sigma(\sigma)\}$ for contractions $f : \Sigma \to \Sigma' \uplus \{\varepsilon\}$ and closed systems σ of $\mu^{\downarrow,1}_\Sigma(\Diamond, \Diamond)$ such that $\mathrm{tree}(\tau)$ is f-good for all $\tau \in \mathrm{L}_\Sigma(\sigma)$.*

4. Safety fragment

Let ITCANT denote the extension of ITCA by nondeterministic transfers. Such an instruction $\langle \mathtt{transf}, c, C\rangle$ transfers the value of counter c to the counters in set C, distributing it among the latter nondeterministically. (Separate zero-test instructions are no longer necessary, since $\langle \mathtt{transf}, c, \emptyset\rangle$ has the same effect as $\langle \mathtt{ifzero}, c\rangle$.) Safety ITCANT are automata obtained from ITCANT by omitting the Büchi acceptance mechanism, so that there is no requirement on infinite sequences of transitions for a run to be accepting. The following two results are obtained from the proof of Theorem 9, and using well-quasi-orderings.

Theorem 13 *For each safety $\mathcal{A}$ in ATRA$^1(\Diamond, \Diamond)$, a safety ITCANT $\mathcal{C}_\mathcal{A}$ with the same alphabet, such that $\mathrm{L}(\mathcal{C}_\mathcal{A}) = \{\mathrm{tree}(\tau) : \tau \in \mathrm{L}(\mathcal{A})\}$, is computable in polynomial space.*

Theorem 14 *Nonemptiness of safety ITCANT is decidable.*

A consequence of Theorems 3, 13 and 14 is decidability of satisfiability for the safety fragment of $\mu^{\downarrow,1}(\Diamond, \Diamond)$. Non-elementarity is shown by encoding as data trees computations of Minsky machines of size k whose counters are bounded by a tower of exponentiations of height k. The data trees involve iterated balanced binary trees which witness the boundedness. Note that, by Theorem 2(a) and from the proof of Theorem 2(b), dropping the safety restriction, or adding $\Box$, $\boxminus$ or one more register, each cause satisfiability to become Π^0_1-hard.

Theorem 15 *Satisfiability for safety $\mu^{\downarrow,1}(\Diamond, \Diamond)$ is decidable and not elementary.*

The final result in this section is obtained using Theorem 3 and the proofs of Theorems 9, 13, 14 and 2(a). Non-primitive recursiveness holds already for validity. From decidability of refinement, we have decidability of satisfiability of arbitrary Boolean combinations of safety systems.

Theorem 16 *Refinement for safety $\mu^{\downarrow,1}(\Diamond, \Diamond)$ is decidable and not primitive recursive.*

5. First-order logic

In this section, we show that there are two-way translations between first-order logics with 2 variables over data trees which were considered in [4] and certain fragments of CTL with freeze quantification and one register. The main result in [4] is that finitary satisfiability for FO$^2(+1, \sim)$ is in 3NEXPTIME. By Theorem 17(b) below, it follows that finitary satisfiability for the fragment SCTL$^{\downarrow,1}(+1)$ of $\mu^{\downarrow,1}(\Diamond, \Box, \Diamond, \boxminus)$ also is in 3NEXPTIME. Recall that, in Corollary 10 above, we established decidability of finitary satisfiability for $\mu^{\downarrow,1}(\Diamond, \Diamond)$. However, by the proof of [11, Theorem 17], finitary satisfiability for $\mu^{\downarrow,1}(\Diamond, \Diamond)$ extended by the operator $\downarrow_1 \mathtt{EXXF}$ (see below) is Σ^0_1-hard.

As in [4], let FO$^2_\Sigma(+1, <, \sim)$ denote first-order logic whose models are data trees with alphabet Σ, and which has 2 variables which range over tree nodes, unary predicates a for each $a \in \Sigma$, and binary predicates $\triangledown$, $\triangleright$, $\triangledown^+$, $\triangleright^+$ and $\sim$. An atomic formula $a(x)$ is true iff the label at node x is a. The predicates $\triangledown$ and $\triangleright$ are interpreted by the "child" and "next sibling" relations, the predicates $\triangledown^+$ and $\triangleright^+$ by the transitive closures of those two relations, and the predicate $\sim$ by the "equality of data labels" relation. FO$^2_\Sigma(+1, \sim)$ denotes the fragment without $\triangledown^+$ and $\triangleright^+$ predicates, and FO$^2_\Sigma(<, \sim)$ the fragment without $\triangledown$ and $\triangleright$ predicates. When 'Σ' is omitted, we consider all finite alphabets. Writing $\phi(x)$ means that the formula ϕ contains free occurences of at most the variable x.

Let SCTL$^{\downarrow,1}_\Sigma(+1, <)$ be the following "simple" fragment of CTL with freeze quantification and one register:

- the atomic formulae are $\top$, $\uparrow_1$, and a for $a \in \Sigma$;
- there are Boolean operators $\neg$ and $\wedge$, and temporal operators $\mathtt{Q}\,\mathtt{L}$ for $\mathtt{Q} \in \{\triangledown\mathtt{E}, \triangle, \triangleright, \triangleleft\}$ and $\mathtt{L} \in \{\mathtt{X}, \mathtt{XXF}\}$;
- each occurence of a temporal operator is immediately preceded by the freeze quantifier $\downarrow_1$.

As shown in Section 2.2, $\mathrm{SCTL}^{\downarrow,1}_{\Sigma}(+1,<)$ is contained in $\mu^{\downarrow,1}_{\Sigma}(\circledcirc, \circledast, \circleddash, \circledcirc)$. Let $\mathrm{SCTL}^{\downarrow,1}_{\Sigma}(+1)$ denote the subfragment with temporal operators $\triangledown\mathtt{EX}$, $\triangle\mathtt{X}$, $\triangleright\mathtt{X}$, $\triangleleft\mathtt{X}$ and $\mathtt{EXXF}$, where $\mathtt{EXXF}\phi$ is equivalent to $(\triangledown\mathtt{EXXF}\phi) \vee (\triangle\mathtt{XXF}\phi) \vee (\triangleright\mathtt{XXF}\phi) \vee (\triangleleft\mathtt{XXF}\phi)$. Let $\mathrm{SCTL}^{\downarrow,1}_{\Sigma}(<)$ denote the subfragment with temporal operators $\triangledown\mathtt{EXF}$, $\triangle\mathtt{XF}$, $\triangleright\mathtt{XF}$ and $\triangleleft\mathtt{XF}$.

Let a formula $\phi(x)$ of $\mathrm{FO}^2_{\Sigma}(+1,<,\sim)$ and a closed system σ of $\mu^{\downarrow}_{\Sigma}(\circledcirc, \circledast, \circleddash, \circledcirc)$ with no free occurences of registers be equivalent iff, for each data tree τ with alphabet Σ and node n, we have $\tau \models_{[x \mapsto n]} \phi$ iff $n \in [\![\sigma]\!]^{\tau}_{\emptyset,\emptyset}(V)$, where V is the last left-hand side in σ. The result below is proved by generalising the translations in the proof of [11, Proposition 6], which is for first-order logic with two variables over data words and linear temporal logic with one register.

Theorem 17 *Suppose $\emptyset \neq \mathcal{R} \subseteq \{+1,<\}$. (a) For each formula $\phi(x)$ of $\mathrm{FO}^2_{\Sigma}(\mathcal{R},\sim)$, an equivalent sentence $\psi^{\Sigma}_{\phi,x}$ of $\mathrm{SCTL}^{\downarrow,1}_{\Sigma}(\mathcal{R})$ is computable in polynomial space. (b) For each sentence ψ of $\mathrm{SCTL}^{\downarrow,1}_{\Sigma}(\mathcal{R})$, an equivalent formula $\phi^{\Sigma}_{\psi}(x)$ of $\mathrm{FO}^2_{\Sigma}(\mathcal{R},\sim)$ is computable in logarithmic space.*

6. XPath

In this section, we first describe how XML documents and DTDs can be represented by data trees and tree automata. We then introduce a forward fragment of XPath, and a safety subfragment. By translating XPath queries to systems of the alternation-free modal μ-calculus with freeze quantification, and applying results in Sections 3 and 4, we establish that finitary satisfiablity for forward XPath and satisfiability for safety forward XPath are decidable.

XML trees. Suppose Σ is a finite set of element types, Σ' is a finite set of attribute names, and Σ and Σ' are disjoint. An XML document [6] is an unranked ordered tree whose every node n is labelled by some $\mathtt{type}(n) \in \Sigma$ and by a datum for each element of some $\mathtt{atts}(n) \subseteq \Sigma'$. Motivated by processing of XML streams (cf., e.g., [27]), we do not restrict our attention to finite XML documents.

Concerning the data in XML documents, we shall consider only the equality predicate between data labels. Equality comparisons with constants are straightforward to encode using additional attribute names. Therefore, similarly as in [4], we represent an XML document by a data tree with alphabet $(\Sigma \cup \Sigma') \uplus \{d, r\}$, where each node n is represented by a node which is labelled by $\mathtt{type}(n)$, whose first $|\mathtt{atts}(n)|$ children are labelled by the elements of $\mathtt{atts}(n)$, and which has two children labelled by d and r. The only child of the d (resp., r) child of n is the node which represents the first child (resp., next sibling) of n (if any). Equalities between data labels are represented by the equivalence relation between nodes which are labelled by attribute names. We say that such a data tree is an *XML tree*.

Following [2, 4], we assume without loss of generality that document type definitions (DTDs) [6] are given as regular tree languages. More precisely, we consider a DTD to be a nondeterministic top-down tree automaton $\mathcal{T}$ with alphabet $(\Sigma \cup \Sigma') \uplus \{d, r\}$ and without ε-transitions. Such automata can be defined by omitting counters and ε-transitions from ITCA (cf. Section 2.4). We say that $\mathcal{T}$ is *safety* iff every location of $\mathcal{T}$ is Büchi. An XML tree τ as above is regarded to satisfy $\mathcal{T}$ iff $\mathtt{tree}(\tau)$ is accepted by $\mathcal{T}$.

Fragments of XPath. The fragment of XPath [7] below contains all operators commonly found in practice and was considered in [2, 17]. The grammars of queries p and qualifiers q are mutually recursive. The element types a and attribute names a' range over Σ and Σ', respectively.

$$\begin{array}{rcl} p & ::= & \varepsilon \mid \triangledown \mid \triangle \mid \triangleright \mid \triangleleft \mid \\ & & \triangledown^* \mid \triangle^* \mid \triangleright^* \mid \triangleleft^* \mid \\ & & p/p \mid p \cup p \mid p[q] \end{array} \qquad \begin{array}{rcl} q & ::= & \neg q \mid q \wedge q \mid p \mid a \mid \\ & & p/@a' = p/@a' \mid \\ & & p/@a' \neq p/@a' \end{array}$$

We say that a query is *forward* iff:

- it does not contain $\triangle$, $\triangleleft$, $\triangle^*$, $\triangleleft^*$;
- for every subqualifier of the form $p_1/@a'_1 \bowtie p_2/@a'_2$, we have that $p_1 = \varepsilon$ and that p_2 is of the form ε or $\triangledown/p'_2$ or $\triangleright/p'_2$.

A *safety* query is one in which each occurence of $\triangledown$, $\triangledown^*$ or $\triangleright^*$ is under an odd number of negations.

The semantics of queries and qualifiers is standard (cf., e.g., [17]). We write the satisfaction relations as $\tau, n, n' \models p$ and $\tau, n \models q$, where τ is an XML tree over Σ and Σ' with root n_R, and n and n' are Σ-labelled nodes. We say that τ satisfies p iff $\tau, n_R, n' \models p$ for some n'.

Example 18 Suppose $a'_1, a'_2 \in \Sigma'$. The forward query $p_{a'_1,a'_2} = \triangledown^*[\varepsilon/@a'_1 = (\triangledown/\triangledown^*)/@a'_2]$ is satisfied by Σ-labelled nodes n_0 and n_1 iff n_1 is equal to or a descendant of n_0 and there exists a descendant n_2 of n_1 such that the value of attribute a'_1 at n_1 is equal to the value of attribute a'_2 at n_2. The safety forward query $\varepsilon[\neg p_{a'_1,a'_2}]$ is satisfied by an XML tree over Σ and Σ' iff the value of a'_1 at a node is never equal to the value of a'_2 at a descendant. □

Suppose a query p and a DTD $\mathcal{T}$ are over the same element types and attribute names. We say that p is satisfiable relative to $\mathcal{T}$ iff there exists an XML tree which satisfies p and $\mathcal{T}$. Finitary satisfiability restricts to finite XML trees.

Complexity of satisfiability. For a system σ and a variable W, writing $\sigma(W)$ means that σ contains free occurences of at most the variable W and no free occurences of registers. Let a query p over element types Σ and attribute names Σ' be equivalent to a system $\sigma(W)$ of $\mu^{\downarrow}_{(\Sigma\cup\Sigma')\uplus\{d,r\}}(\Diamond, \Diamond\!\!\!\!-, \Diamond\!\!\!\!=, \Diamond\!\!\!\!\!=)$ iff, for every XML tree τ over Σ and Σ', Σ-labelled node n, and set M of nodes, we have $n \in [\![\sigma]\!]^{\tau}_{[W\mapsto M],\emptyset}(V)$ iff there exists $m \in M$ with $\tau, n, m \models p$, where V is the last left-hand side in σ.

Theorem 19 *For each forward query p over Σ and Σ', an equivalent system $\sigma_p^{\Sigma,\Sigma'}(W)$ of $\mu^{\downarrow,1}_{(\Sigma\cup\Sigma')\uplus\{d,r\}}(\Diamond, \Diamond\!\!\!=)$ is computable in logarithmic space. Safety is preserved.*

The following decidability results are obtained from Theorems 19, 3, 9, 8, 13 and 14.

Theorem 20 *(a) Finitary satisfiability for forward XPath and arbitrary DTDs is decidable. (b) Satisfiability for safety forward XPath and safety DTDs is decidable.*

7. Conclusion

The principal objective of the paper was to analyse the complexity of satisfiability for fragments of the alternation-free modal μ-calculus with freeze quantification, with and without restricting to finite data trees. An overview of the main results is in the following table, where 'R, not PR' means 'decidable and not primitive recursive', and 'R, not EL' means 'decidable and not elementary'.

	fin. sat.	satisfiability
safety $\mu^{\downarrow,1}(\Diamond, \Diamond\!\!\!=)$	R, not PR	R, not EL
Boolean closure of safety $\mu^{\downarrow,1}(\Diamond, \Diamond\!\!\!=)$	R, not PR	R, not PR
$\mu^{\downarrow,1}(\Diamond, \Diamond\!\!\!=)$	R, not PR	Π^0_1-complete
$\mu^{\downarrow,1}(\Diamond, \Diamond)$, $\mu^{\downarrow,1}(\Diamond\!\!\!-, \Diamond\!\!\!\!=)$, $\mu^{\downarrow,2}(\Diamond)$, $\mu^{\downarrow,2}(\Diamond\!\!\!-)$	Σ^0_1-complete	Σ^1_1-complete

References

[1] R. Alur and T. Henzinger. A really temporal logic. *JACM*, 41(1):181–204, 1994.

[2] M. Benedikt, W. Fan, and F. Geerts. XPath satisfiability in the presence of DTDs. In *PODS*, pages 25–36. ACM, 2005.

[3] H. Björklund and M. Bojańczyk. Bounded depth data trees. In *ICALP*, LNCS. Springer, 2007.

[4] M. Bojańczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. In *PODS*, pages 10–19. ACM, 2006.

[5] M. Bojańczyk, A. Muscholl, T. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS*, pages 7–16. IEEE, 2006.

[6] T. Bray, J. Paoli, and C. Sperberg-McQueen. Extensible markup language (XML) 1.0. W3C Recommendation, 1998.

[7] J. Clark and S. DeRose. XML path language (XPath). W3C Recommendation, 1999.

[8] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM TOPLAS*, 8(2):244–263, 1986.

[9] C. David. Mots et données infinies. Master's thesis, LIAFA, Université Paris 7, 2004.

[10] P. deGroote, B. Guillaume, and S. Salvati. Vector addition tree automata. In *LICS*, pages 64–73. IEEE, 2004.

[11] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. In *LICS*, pages 17–26. IEEE, 2006.

[12] S. Demri, R. Lazić, and D. Nowak. On the freeze quantifier in Constraint LTL: decidability and complexity. *I & C*, 205(1):2–24, 2007.

[13] E. A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional mu-calculus. In *LICS*, pages 267–278. IEEE, 1986.

[14] K. Etessami, M. Vardi, and T. Wilke. First-order logic with two variables and unary temporal logic. *I & C*, 179(2):279–295, 2002.

[15] A. Finkel and P. Schnoebelen. Well-structured transitions systems everywhere! *TCS*, 256(1–2):63–92, 2001.

[16] T. French. Quantified propositional temporal logic with repeating states. In *TIME-ICTL*, pages 155–165. IEEE, 2003.

[17] F. Geerts and W. Fan. Satisfiability of XPath queries with sibling axes. In *DBPL*, volume 3774 of *LNCS*, pages 122–137. Springer, 2005.

[18] S. Hallé, R. Villemaire, and O. Cherkaoui. CTL model checking for labelled tree queries. In *TIME*, pages 27–35. IEEE, 2006.

[19] M. Kaminski and N. Francez. Finite-memory automata. *TCS*, 134(2):329–363, 1994.

[20] M. Kaminski and T. Tan. Tree automata over infinite alphabets. Poster at CIAA, 2006.

[21] O. Kupferman and M. Vardi. Weak alternating automata are not that weak. *ACM TOCL*, 2(3):408–429, 2001.

[22] R. Lazić. Safely freezing LTL. In *FSTTCS*, volume 4337 of *LNCS*, pages 381–392. Springer, 2006.

[23] A. Lisitsa and I. Potapov. Temporal logic with predicate λ-abstraction. In *TIME*, pages 147–155. IEEE, 2005.

[24] M. Minsky. *Computation, Finite and Infinite Machines*. Prentice Hall, 1967.

[25] F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM TOCL*, 5(3):403–435, 2004.

[26] P. Odifreddi. *Classical Recursion Theory II*. Elsevier, 1999.

[27] D. Olteanu, T. Furche, and F. Bry. An efficient single-pass query evaluator for XML data streams. In *SAC*, pages 627–631. ACM, 2004.

[28] J. Ouaknine and J. Worrell. On Metric temporal logic and faulty Turing machines. In *FOSSACS*, volume 3921 of *LNCS*, pages 217–230. Springer, 2006.

[29] P. Schnoebelen. Verifying lossy channel systems has non-primitive recursive complexity. *IPL*, 83(5):251–261, 2002.

[30] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.

[31] I. Walukiewicz. Pushdown processes: Games and model-checking. *I & C*, 164(2):234–263, 2001.