Expressiveness of a spatial logic for trees

Iovka Boneva, Jean-Marc Talbot, Sophie Tison

Expressiveness of Spatial Logic for Trees

Iovka Boneva, Jean-Marc Talbot and Sophie Tison
LIFL - UMR CNRS 8022 and INRIA Futurs - MOSTRARE project
59655 Villeneuve d'Ascq Cédex, France
{boneva, talbot, tison}@lifl.fr

Abstract

In this paper we investigate the quantifier-free fragment of the TQL logic proposed by Cardelli and Ghelli. The TQL logic, inspired from the ambient logic, is the core of a query language for semistructured data represented as unranked and unordered trees. The fragment we consider here, named STL, contains as main features spatial composition and location as well as a fixed point construct. We prove that satisfiability for STL is undecidable. We show also that STL is strictly more expressive than the Presburger monadic second-order logic (PMSO) of Seidl, Schwentick and Muscholl when interpreted over unranked and unordered edge-labelled trees. We define a class of tree automata whose transitions are conditioned by arithmetical constraints; we show then how to compute from a closed STL formula a tree automaton accepting precisely the models of the formula. Finally, still using our tree automata framework, we exhibit some syntactic restrictions over STL formulae that allow us to capture precisely the logics MSO and PMSO.

1 Introduction

Spatial logics allow one to express properties about structures such as trees [8], graphs [6, 14] and heaps [19]. The main ingredient of spatial logics is an operator called composition (or separation). This operator permits compositional reasoning over concurrent and mobile processes [10, 4] or over imperative programs with shared mutable data structures [18].

The TQL logic [7, 8] is the core of a query language for semistructured data represented as unranked and unordered trees. The TQL logic is based on the static fragment of the ambient logic: it contains spatial primitives, least fixed point operator and quantification over labels and trees. Spatial connectives are location $a[\phi]$, satisfied by trees having a single edge starting from the root that is labelled with a and leads to a subtree satisfying the formula ϕ and composition $\phi \mid \phi'$ satisfied by trees which can be obtained by merging the roots of two trees, one satisfying ϕ and the other one ϕ'. The least fixed point operator considers formulae as mappings over sets of trees.

An important question about logics is the decidability of satisfiability. It has been shown in [11] that satisfiability is undecidable for a fragment of the ambient logic contained in TQL. The use of quantification over names is crucial for this result. However, decidable fragments of the logic TQL could be useful in several domains: for constructing type systems for semistructured data as the one proposed in [5] or for testing query correctness, query containment and defining constraints, as suggested in [8].

The logic TL introduced by Dal Zilio *et al.* in [12, 13] is also inspired by the ambient logic. This logic is quantifier-free and equipped with a restricted form of recursion but it allows an additional spatial primitive (composition adjunct). However, this new construct adds no expressiveness to the logic. The logic TL can be encoded into the so called sheaves automata [12], which are automata whose transitions are conditioned by Presburger formulae. As emptiness is decidable for sheaves automata, satisfiability is decidable for the logic TL. Sheaves automata are the first proposal of a class of tree automata for ambient-based logics. It is not clear whether the restriction of recursion in TL is necessary for decidability. Additionally, it is not known whether sheaves-like automata could be defined for larger fragments of TQL.

Another fundamental question is the expressiveness of a logic. A well-known logic used to express properties about trees is the monadic second-order logic (MSO). It is known that the TQL logic can express some counting properties in trees that can not be defined in MSO; the formula $\mu\xi.(a[\xi] \mid b[\xi] \mid \xi \vee \mathbf{0})$ expresses that any node has as many outgoing edges labelled with a as outgoing edges labelled with b. Recently, Seidl *et al.* introduced in [20] the Presburger monadic second order (PMSO) as an extension of MSO with some counting constraints allowing one to express relationship between the number of children of a given node satisfying some property. An example of such

a constraint is "as many children labelled with a as children labelled with b". Seidl *et al.* also introduced so called Presburger tree automata and showed equivalence between these automata and the logic PMSO. Presburger automata are quite similar to sheaves automata as they both have Presburger arithmetic formulae as conditions for transitions. This similarity suggests that there is probably a strong relation between fragments of the TQL logic and PMSO logic, but the precise relationship is unknown yet.

In this paper we focus on the quantification-free fragment of the TQL logic that we call spatial tree logic (STL for short). We study the satisfiability problem as well as the expressiveness of this logic. In particular, we compare STL with MSO and PMSO logics when interpreted over unranked and unordered trees.

We show that satisfiability is undecidable for STL and that STL is strictly more expressive than the logic PMSO when interpreted over unranked and unordered trees. We also identify two syntactic fragments of STL, denoted gSTL and pgSTL, that correspond precisely to PMSO and to MSO logics respectively. The equivalence is shown using a new tree automata framework for STL: we define a general class of tree automata in the spirit of Presburger and sheaves automata that use some arithmetical constraints as conditions for the transition relation. By restricting these arithmetical constraints, we obtain subclasses of automata equivalent to the logics gSTL and pgSTL respectively.

The paper is organised as follows: Section 2 contains definitions for the tree model we consider and for the logics STL, MSO and PMSO; in Section 3 we introduce so called automata with arithmetical constraints and we define two subclasses of these automata that are equivalent to the logics MSO and PMSO respectively. We prove in Section 4 undecidability of satisfiability for STL; Section 5 describes the construction of an automaton equivalent to some STL formula. Finally, in Section 6 we define the fragments gSTL and pgSTL of STL and we use automata with arithmetical constraints to show equivalence between these two fragments and the logics MSO and PMSO respectively.

2 Tree model and logics : definitions

Let Λ be a finite set of labels. All through this paper, by tree we mean a finite, unranked and unordered tree with edges labelled from Λ.

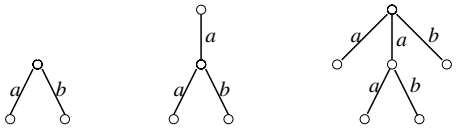
We consider two different representations of trees namely as nested multisets and as logical structures.

Nested multisets This representation, introduced in [7], is quite natural for unranked and unordered trees.

Let us denote $\{\!\{\}\!\}$ the empty multiset and $\uplus$ the multiset union. The set Tree is the least set containing the empty multiset $\{\!\{\}\!\}$ and such that if t', t'' belong to Tree and a is a label from Λ, then $t' \uplus t''$ and $\{\!\{a[t']\}\!\}$ belong to Tree as well. Elements of multisets, also called *tree elements*, are always of the form $a[t]$ for some label a and some tree t. The set of all tree elements is denoted ETree.

We write $\{\!\{e_1, \ldots, e_n\}\!\}$ for the multiset containing the tree elements $e_1, \ldots, e_n$. If t is a multiset, then $t(e)$ denotes the multiplicity (*ie* the number of occurrences) of the element e in t. For any natural n and any tree element e, we denote by $n \cdot e$ the tree $\{\!\{\underbrace{e, \ldots, e}_{n \text{ times}}\}\!\}$.

Below are depicted graphical representations of the trees (in this order) $t = \{\!\{a[\{\!\{\}\!\}],\ b[\{\!\{\}\!\}]\}\!\}$, $t' = \{\!\{\ a[\{\!\{a[\{\!\{\}\!\}],\ b[\{\!\{\}\!\}]\}\!\}]\ \}\!\}$ and $t'' = \{\!\{a[\{\!\{a[\{\!\{\}\!\}],\ b[\{\!\{\}\!\}]\}\!\}],\ a[\{\!\{\}\!\}],\ b[\{\!\{\}\!\}]\}\!\}$. Note that $t' = \{\!\{a[t]\}\!\}$ and $t'' = t \uplus t'$.



Logical structure Let σ be the signature $\{\mathsf{lab}_a \mid a \in \Lambda\} \cup \{<\}$, where the lab_a are unary predicates and $<$ is a binary predicate. A tree τ induces a finite σ-structure $\langle E^\tau, \{\mathsf{lab}_a^\tau \mid a \in \Lambda\}, <^\tau \rangle$ where E^τ is the finite set of edges of τ, lab_a^τ (for $a \in \Lambda$) is the labelling relation, that is $\mathsf{lab}_a^\tau(u)$ holds if the edge u in τ is labelled with a and $<^\tau$ is the predecessor relation, *ie* $u <^\tau u'$ holds for the edges u, u' in τ if the destination of the edge u and the source of the edge u' coincide.

There is a natural correspondence between trees as nested multisets and (isomorphism-closed classes of) trees as logical structures.

We consider here the spatial tree logic, denoted STL, which is the quantifier-free fragment of the TQL logic [7].

Syntax Assume a countable set of recursion variables ranged over by ξ, ξ'. STL formulae are defined by the following grammar (for $\alpha \subseteq \Lambda$):

$$\phi ::= \mathbf{0} \mid \alpha[\phi] \mid \phi \,|\, \phi \mid \top \mid \neg\phi \mid \phi \vee \phi \mid \xi \mid \mu\xi.\phi$$

To guarantee the existence of least fixed points (μ operator) we impose a syntactic restriction for formulae $\mu\xi.\phi$: occurrences of the recursion variable ξ must occur under an even number of negations ($\neg$) in ϕ.

Some derived operators $\overline{\mathbf{0}}$, $\overline{\alpha}[\phi]$, $\phi \,\|\, \phi'$, $\phi \wedge \phi'$, $\nu\xi.\phi$ can be defined using negation as: $\overline{\mathbf{0}} \stackrel{\text{def}}{=} \neg\mathbf{0}$, $\overline{\alpha}[\phi] \stackrel{\text{def}}{=} (\Lambda \setminus \alpha)[\phi]$,

$\phi \,\|\, \phi' \stackrel{\text{def}}{=} \neg(\neg\phi \,|\, \neg\phi')$, $\phi \wedge \phi' \stackrel{\text{def}}{=} \neg(\neg\phi \vee \neg\phi')$ and $\nu\xi.\phi \stackrel{\text{def}}{=} \neg(\mu\xi.\neg\phi\langle\xi \leftarrow \neg\xi\rangle)$, where $\phi\langle\xi \leftarrow \neg\xi\rangle$ is the formula ϕ in which occurrences of the recursion variable ξ have been replaced with $\neg\xi$.

The notions of free variables and closed formulae are defined as usual. In the sequel of this paper, we assume that in STL formulae, distinct occurrences of the binders μ and ν bind distinct variables. We use κ to denote either of the symbols μ or ν. For some label a and some formula ϕ, $a[\phi]$ stands for the formula $\{a\}[\phi]$.

Semantics The semantics of STL is given by an interpretation associating a set of trees with any formula. Let δ be a valuation associating a set of trees with any free recursion variable. The interpretation of the formula ϕ under the mapping δ, denoted $[\![\phi]\!]_\delta$, is recursively defined by:

$$\begin{array}{ll}
[\![\mathbf{0}]\!]_\delta & = \{\{\!|\}\!\} \\
[\![\alpha[\phi]]\!]_\delta & = \{\{\!|a[t]|\!\} \mid a \in \alpha, t \in [\![\phi]\!]_\delta\} \\
[\![\phi \,|\, \phi']\!]_\delta & = \{t \uplus t' \mid t \in [\![\phi]\!]_\delta, t' \in [\![\phi']\!]_\delta\} \\
[\![\top]\!]_\delta & = \mathsf{Tree} \\
[\![\neg\phi]\!]_\delta & = \mathsf{Tree} \setminus [\![\phi]\!]_\delta \\
[\![\phi \vee \phi']\!]_\delta & = [\![\phi]\!]_\delta \cup [\![\phi']\!]_\delta \\
[\![\xi]\!]_\delta & = \delta(\xi) \\
[\![\mu\xi.\phi]\!]_\delta & = \bigcap\{S \subseteq \mathsf{Tree} \mid S \supseteq [\![\phi]\!]_{\delta[\xi \mapsto S]}\}
\end{array}$$

where $\delta[\xi \mapsto S]$ stands for the valuation δ identical to δ except for the variable ξ which is mapped to the set S.

The syntax of STL can be enriched with the star operator: if ϕ is a STL formula, then so is $\phi*$. The interpretation of $\phi*$ is $[\![\phi*]\!]_\delta = \bigcup_{n\in\mathbb{N}}\{t_1 \uplus \ldots \uplus t_n \mid \forall i \in 1..n.\, t_i \in [\![\phi]\!]_\delta\}$. The star operator does not add expressive power to the logic as, for any valuation δ, $[\![\phi*]\!]_\delta = [\![\mu\xi.(\phi \,|\, \xi \vee \mathbf{0})]\!]_\delta$.

A set of trees T is said to be STL *definable* if there exists some closed STL formula ϕ such that $T = [\![\phi]\!]$.

System of fixed point equations We present here an alternative representation of the semantics of a closed STL formula as the solution of a system of fixed point equations.

Following [1], a *system of fixed point equations* Σ over the variables $\xi_1, \ldots, \xi_n$ is a sequence $\xi_1 \stackrel{\kappa}{=} f_1(\xi_1, \ldots, \xi_n), \ldots, \xi_n \stackrel{\kappa}{=} f_n(\xi_1, \ldots, \xi_n)$ where the f_i denote functionals that are intended to be interpreted as monotonic mappings over some domain. The sequence of variables $\xi_1, \ldots, \xi_n$ is denoted $\mathsf{Vars}(\Sigma)$ and ξ_n, the last variable in $\mathsf{Vars}(\Sigma)$, is denoted $\mathsf{last}(\Sigma)$.

We interpret here equations over the complete lattice structure induced by set inclusion $(\wp(\mathsf{Tree}), \vee, \wedge)$, where $\wp(\mathsf{Tree})$ is the power set of Tree, $\vee$ is set union and $\wedge$ is set intersection. We consider the following monotonic operations over $\wp(\mathsf{Tree})$: the constant $\mathbf{0}$ is interpreted as $\{\{\!|\}\!\}$, $\overline{\mathbf{0}}$ as $\mathsf{Tree} \setminus \{\{\!|\}\!\}$ and $\top$ is the greatest element of $\wp(\mathsf{Tree})$, *ie* Tree; for $T, T' \in \wp(\mathsf{Tree})$ and $\alpha \subseteq \Lambda$, the unary operator $\alpha[T]$ is interpreted as the set $\{a[t] \mid a \in \alpha, t \in T\}$ and $\overline{\alpha}[T]$ as the set $\{a[t] \mid a \notin \alpha, t \in T\}$. Finally, the binary operator $T \,|\, T'$ is interpreted as the set $\{t'' \mid \exists t, t'. t \in T \text{ and } t' \in T' \text{ and } t'' = t \uplus t'\}$ and $T \,\|\, T'$ as the set $\{t'' \mid \forall t, t'.\ t'' \neq t \uplus t' \text{ or } t \in T \text{ or } t' \in T'\}$.

The solution of Σ over the lattice $(\wp(\mathsf{Tree}), \wedge, \vee)$ is a mapping from $\{\mathsf{Vars}(\Sigma)\}$ into $\wp(\mathsf{Tree})$, denoted $\mathcal{S}_\Sigma$.

A closed STL formula ϕ is said to be in *negation normal form* if it is a negation-free formula built over initial and derived operators. Any closed STL formula can be put in negation normal form by pushing negation deeper inside the formula using derived operators and the fact that $\neg\alpha[\phi']$ is equivalent to $\overline{\mathbf{0}} \,|\, \overline{\mathbf{0}} \vee \mathbf{0} \vee \overline{\alpha}[\top] \vee \alpha[\neg\phi']$.

Let ϕ be a closed STL formula in negation normal form. Assume wlog that $\phi = \mu\xi.\phi'$ for some formula ϕ'. The system of equations $\mathsf{Eq}(\phi)$ is defined inductively on the structure of ϕ as: $\mathsf{Eq}(\psi) =$

$$\begin{cases}
\mathsf{Eq}(\psi'), \xi' \stackrel{\kappa}{=} \mathsf{tr}(\psi') & \text{if } \psi = \kappa\xi'.\psi' \\
\mathsf{Eq}(\psi'), \mathsf{Eq}(\psi'') & \text{if } \psi = \psi' \circ \psi'' \text{ for } \circ \in \{\vee, \wedge, |, \|\} \\
\mathsf{Eq}(\psi') & \text{if } \psi = \alpha[\psi'] \text{ or } \psi = \overline{\alpha}[\psi'] \\
\epsilon & \text{otherwise}
\end{cases}$$

where , composes sequentially two systems of equations, ϵ is the neutral element of this composition and $\mathsf{tr}(\psi)$ is the formula obtained from ψ by replacing in ψ any recursion sub-formula $\kappa\xi'.\psi'$ by the variable ξ'. Note that, by construction of $\mathsf{Eq}(\phi)$, $\mathsf{last}(\mathsf{Eq}(\phi)) = \xi$.

The semantics of STL formulae defined by systems of equations is correct in the sense that

Proposition 1 *For any closed STL formula ϕ in negation normal form, $\mathcal{S}_{\mathsf{Eq}(\phi)}(\mathsf{last}(\mathsf{Eq}(\phi))) = [\![\phi]\!]$.*

Example 1 *Let $\phi = \mu\xi_2.\phi_2$, $\phi_2 = \alpha[\neg\mu\xi_1.\phi_1] \,|\, \xi_2 \vee \mathbf{0}$ and $\phi_1 = \xi_1 \,|\, \neg\xi_2 \vee \mathbf{0}$. The formula ϕ is equivalent to the formula ϕ' in negation normal form defined as $\phi' = \mu\xi_2.\phi'_2$, where $\phi'_2 = \alpha[\nu\xi_1.\phi'_1] \,|\, \xi_2 \vee \mathbf{0}$ and $\phi'_1 = \xi_1 \,\|\, \xi_2 \wedge \overline{\mathbf{0}}$. Therefore, the system of equations $\mathsf{Eq}(\phi)$ is*

$$\xi_1 \stackrel{\nu}{=} \xi_1 \,\|\, \xi_2 \wedge \overline{\mathbf{0}}, \quad \xi_2 \stackrel{\mu}{=} \alpha[\xi_1] \,|\, \xi_2 \vee \mathbf{0}$$

Conversely, with any system of fixed point equations Σ built over Boolean operations $\wedge, \vee$ and the operators $\mathbf{0}$, $\overline{\mathbf{0}}$, $\top$, $\alpha[\,]$, $\overline{\alpha}[\,]$, $|$ and $\|$, one can associate a closed STL formula denoted $\mathsf{form}(\Sigma)$ such that $\mathcal{S}_\Sigma(\mathsf{last}(\Sigma)) = [\![\mathsf{form}(\Sigma)]\!]$. The formula $\mathsf{form}(\Sigma)$ can be effectively constructed: assume $\mathsf{Vars}(\Sigma) = \xi_1, \ldots, \xi_n$ and let for any $i \in 1..n$, $\xi_i \stackrel{\kappa_i}{=} \phi_i$ be the i^{th} equation of Σ. Iteratively, for any i from 1 to n, and for any $j < i$, we replace occurrences of the recursion variable ξ_j in the formula ϕ_i by the formula $\kappa_j\xi_j.\phi_j$. Then $\mathsf{form}(\Sigma)$ is equal to $\kappa_n\xi_n.\phi_n$.

We recall the definition of monadic second-order logic (MSO) for trees and we introduce, following [20], an ex-

tension of MSO called Presburger MSO (PMSO).

MSO Considering the signature σ, MSO formulae are defined by the following syntax:

$$\psi ::= \quad (\mathsf{lab}_a(u))_{a\in\Lambda} \mid u < u' \mid u \in U \mid \neg\psi \mid \psi \wedge \psi \\ \mid \exists u.\psi \mid \exists U.\psi$$

where u and U stand for first order and second order variables respectively.

Semantics of MSO is given by means of a satisfiability relation. Let τ be a tree given as the finite σ-structure $\langle E^\tau, \{\mathsf{lab}_a^\tau \mid a \in \Lambda\}, <^\tau\rangle$, and ρ be a valuation associating elements of E^τ with first order variables and subsets of E^τ with second order variables. Then satisfaction relation for a tree τ under valuation ρ and a MSO formula ψ, denoted $\tau, \rho \models \psi$, is recursively defined on the structure of ψ as:

$\tau, \rho \models \mathsf{lab}_a(u)$	if	$\mathsf{lab}_a^\tau(\rho(u))$ holds in τ
$\tau, \rho \models u < u'$	if	$\rho(u) <^\tau \rho(u')$ holds in τ
$\tau, \rho \models u \in U$	if	$\rho(u)$ belongs to $\rho(U)$
$\tau, \rho \models \neg\psi$	if	$\tau, \rho \not\models \psi$
$\tau, \rho \models \psi \wedge \psi'$	if	$\tau, \rho \models \psi$ and $\tau, \rho \models \psi'$
$\tau, \rho \models \exists u.\psi$	if	$\tau, \rho[u \mapsto e] \models \psi$ for some $e \in E^\tau$
$\tau, \rho \models \exists U.\psi$	if	$\tau, \rho[u \mapsto E] \models \psi$ for some $E \subseteq E^\tau$

PMSO The logic PMSO has been introduced in [20]. The interpretation we give to this logic slightly differs from the original one: Seidl *et al.* consider in [20] a vertex-based MSO whereas we consider here an edge-based MSO.

Presburger arithmetic is the first order theory over the structure $\langle \mathbb{N}, +, = \rangle$ of natural numbers with addition and equality. Let x be a numerical variable and n be a natural number. Presburger formulae p are defined as:

$$p ::= v = v \mid \neg p \mid p \wedge p \mid \exists x.p \quad \text{with } v ::= v{+}v \mid x \mid n$$

Let η be an assignment for the free variables occurring in a Presburger formula p, extended canonically over numerical terms v. Satisfaction relation for a Presburger formula p and an assignment η is inductively defined as:

$\eta \models v = v'$	if $\eta(v) = \eta(v')$
$\eta \models \neg p$	if $\eta \not\models p$
$\eta \models p \wedge p'$	if $\eta \models p$ and $\eta \models p'$
$\eta \models \exists x.p$	if $\eta[x \mapsto n] \models p$ for some natural n

PMSO is defined by extending MSO with a new kind of atomic formulae u/p where p is a Presburger formula constructed over the set of variables $\{\#U \mid U$ is a second order variable$\}$. Satisfaction relation is extended to these atomic formulae: for some tree τ with domain E^τ and some valuation ρ, let η be the valuation associating the cardinality of the set $\rho(U) \cap \{e \mid \rho(u) <^\tau e\})$ with each variable $\#U$ occurring in the Presburger formula p. Then $\tau \models_\rho u/p$ if $\eta \models p$.

A set of trees T is called MSO *definable* (resp. PMSO *definable*) if there exists some MSO (resp. PMSO) sentence ψ such that $\tau \models \psi$ iff $\tau \in T$.

3 Automata with arithmetical constraints

We present a class of automata, very close to Presburger tree automata [20], but adapted to edge-labelled trees and for which Presburger formulae are replaced by more general arithmetical constraints. Our automata are also close to sheaves automata [13, 12] and multitree automata [16]. We also relate these automata to the logics PMSO and MSO.

In the following, for any sets S, T we denote T^S the set of mappings from S into T. Moreover, we freely identify elements from $\mathbb{N}^S$ with multisets over the set S.

Definition of automata An *automaton with arithmetical constraints* (called simply automaton in the sequel) is a tuple (Λ, Q, Δ, F) where Λ is a finite set of symbols, Q is a finite set of states, Δ is a mapping associating with each state q in Q a pair from $(\wp(\mathbb{N}^Q), \wp(\Lambda))$ and finally, $F \subseteq \mathbb{N}^Q$ is the acceptance condition.

Let $A = (\Lambda, Q, \Delta, F)$ be an automaton. For any state q in Q and for any set of mappings $N \subseteq \mathbb{N}^Q$, we define the sets $\mathcal{L}(q) \subseteq \mathsf{ETree}$ and $\mathcal{L}(N) \subseteq \mathsf{Tree}$. $\mathcal{L}(q)$ is the set of tree elements accepted in the state q of the automaton and $\mathcal{L}(N)$ is the set of trees accepted by the arithmetical condition N. These two sets are recursively defined as follows:

- $\mathcal{L}(q)$ is the set of tree elements $\alpha_q[\mathcal{L}(N_q)]$, where α_q is the set of labels and N_q is the set of mappings from $\mathbb{N}^Q$ such that $\Delta(q) = (N_q, \alpha_q)$;
- $\mathcal{L}(N)$ is the set of trees $t = \{\!\{e_1, \ldots, e_k\}\!\}$ for which there exists a multiset of states $\{\!\{q^1, \ldots, q^k\}\!\}$ in N such that $e_i \in \mathcal{L}(q^i)$ for all $i \in 1..k$.

Then the language accepted by the automaton A, written $\mathcal{L}(A)$, is the set of trees $\mathcal{L}(F)$.

Subclasses of automata We define first star-free and semilinear sets of mappings by analogy with star-free and semilinear sets of vectors.

For $\mathbf{n}, \mathbf{n}' \in \mathbb{N}^Q$ and $\lambda \in \mathbb{N}$, we define the mappings $\mathbf{n}{+}\mathbf{n}'$ and $\lambda\mathbf{n}$ as: $(\mathbf{n}{+}\mathbf{n}')(q) = \mathbf{n}(q){+}\mathbf{n}'(q)$ and $(\lambda\mathbf{n})(q) = \lambda\mathbf{n}(q)$ for all $q \in Q$. Let $\mathbf{p}_1, \ldots, \mathbf{p}_r$ and $\mathbf{b}$ be mappings in $\mathbb{N}^Q$. We denote $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_r)$ the set of mappings

$$\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_r) = \mathbf{b} + \sum_{i\in 1..r, \lambda_i \in \mathbb{N}} \lambda_i \mathbf{p}_i$$

A set of mappings of the form $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_r)$ is called a *linear set*; the mapping $\mathbf{b}$ is its basis and $\mathbf{p}_1, \ldots, \mathbf{p}_r$ are its periods.

A set of mappings $N \subseteq \mathbb{N}^Q$ is *star-free* if it can represented as a finite union of linear sets $\mathsf{Lin}(\mathbf{b}, \mathbf{p}_1, \ldots, \mathbf{p}_r)$ for which all the periods $\mathbf{p}_i$ are unit mappings, *ie* $\exists q \in Q, \mathbf{p}_i(q) = 1$ and $\forall q' \neq q, \mathbf{p}_i(q') = 0$.

A set of mappings $N \subseteq \mathbb{N}^Q$ is *semilinear* if it can be represented as a finite union of linear sets.

A *star-free* (resp. *semilinear*) *constrained automaton* is an automaton for which all arithmetical conditions are star-free (resp. semilinear) sets of mappings.

A semilinear (or a star-free) subset of $\mathbb{N}^Q$ is effectively given when it is given as a finite union of linear sets, each of which is represented by a basis and a finite number of periods. A star-free (resp. semilinear) constrained automaton is effectively given if all arithmetical constraints in the automaton are effectively given star-free (resp. semilinear) sets of mappings.

Determinisation and closure under Boolean operations An automaton is *deterministic* if for any states q, q', the set $\mathcal{L}(q) \cap \mathcal{L}(q')$ is empty. For any effectively given star-free (resp. semilinear) constrained automaton, one can construct a deterministic star-free (resp. semilinear) constrained automaton recognising the same language. These two classes of automata are also closed under Boolean operations.

Deciding emptiness For an automaton $A = (\Lambda, Q, \Delta, F)$, deciding emptiness of the set of trees $\mathcal{L}(A)$ can be done under certain conditions. A state of the automaton $q \in Q$ is *reachable* if $\mathcal{L}(q)$ is not empty; in the same way, a set N of mappings from $\mathbb{N}^Q$ is *reachable* if $\mathcal{L}(N)$ is not empty. Hence, N is reachable iff it contains a mapping $\mathbf{n}$ such that for all $q \in Q$, $\mathbf{n}(q) \neq 0$ implies that q is reachable. Note that if N contains the null mapping 0^Q, *ie* the mapping associating 0 with any $q \in Q$, then it is reachable, as in that case $\{\!\{\}\!\}$ belongs to $\mathcal{L}(N)$. On the other hand, a state q is reachable if, for $\alpha \subseteq \Lambda$ and $N \subseteq \mathbb{N}^Q$ such that $\Delta(q) = (N, \alpha)$, the set α is not empty and N is reachable. According to these remarks, Proposition 2 below gives sufficient conditions to make emptiness decidable for automata.

Proposition 2 *Let $A = (\Lambda, Q, \Delta, F)$ be an automaton. Emptiness is decidable if for any state q in Q with $\Delta(q) = (N, \alpha)$, and for any subset of states $Q' \subseteq Q$, one can decide whether the set of mappings N contains some $\mathbf{n}$ such that for any state q'' in Q, $\mathbf{n}(q'') \neq 0$ implies $q'' \in Q'$.*

Conditions from Proposition 2 are fulfilled by effectively given star-free (resp. semilinear) constrained automata. So,

Corollary 3 *Emptiness is decidable for effectively given star-free and semilinear constrained automata.*

Connection with the logics MSO and PMSO

Proposition 4 *A set of trees is accepted by some semilinear constrained automaton iff it is PMSO definable.*

Proof Using the relationship between semilinear sets and solutions of Presburger formulae, the result easily follows from the equivalence of expressiveness of Presburger automata and PMSO logic sentences as established in [20]. □

Proposition 5 ([3]) *A set of trees is accepted by some star-free constrained automaton iff it is MSO definable.*

4 Satisfiability of STL

The satisfiability problem is, given a closed STL formula ϕ, decide whether $[\![\phi]\!]$ is empty. We show here that this problem is undecidable for STL and that STL is more expressive than the logic PMSO.

Theorem 6 *Satisfiability is undecidable for STL.*

Sketch of proof The proof is done using a reduction of emptiness of two-counter machines inspired from the proof of undecidability of emptiness for alternating two-way AC-tree automata [15]. A two-counter machine [17] $\mathcal{M} = \langle Q, q_i, q_f, \Delta \rangle$ is a finite labelled transition system where Q is the set of states, q_i is the initial state, q_f is the final state and Δ is a set of transitions of the form (q, r, q'), where $q, q' \in Q$ and r belongs to the set $\mathsf{Trans} = \bigcup_{j \in 0,1} \{\oplus_j, \ominus_j, 0_j\}$. A *configuration* of the machine $\mathcal{M}$ is a triple (q, n_0, n_1), where q is a state of the machine and n_0, n_1 are naturals called values of the counters. The set of transitions Δ defines the *step relation* $\vdash_{\mathcal{M}}$ over configurations as follows: $(q, n_0, n_1) \vdash_{\mathcal{M}} (q', n_0', n_1')$ if one of the three statements is true for $j, k \in \{0, 1\}$ and $j \neq k$: (i) $n_j' = n_j + 1$, $n_k' = n_k$ and $(q, \oplus_j, q') \in \Delta$; (ii) $n_j' = n_j - 1$, $n_k' = n_k$ and $(q, \ominus_j, q') \in \Delta$ (iii) $n_j = n_j' = 0$, $n_k = n_k'$ and $(q, 0_j, q') \in \Delta$. As usual $\vdash_{\mathcal{M}}^*$ denotes the transitive reflexive closure of $\vdash_{\mathcal{M}}$. The *language* accepted by the two-counter machine $\mathcal{M}$, denoted $\mathcal{L}(\mathcal{M})$, is the set of naturals n such that $(q_i, n, 0) \vdash_{\mathcal{M}}^* (q_f, m_0, m_1)$ for some naturals m_0, m_1. Emptiness of $\mathcal{L}(\mathcal{M})$ is undecidable for an arbitrary two-counter machine [17], as any recursively enumerable set of naturals can be represented as the language of some two-counter machine.

Consider an arbitrary two-counter machine $\mathcal{M} = \langle Q, q_i, q_f, \Delta \rangle$. A tuple (n_0, x_0, n_1, x_1) of $\mathbb{N}^4$ is said to be *correct* if $n_0 \geq x_0$ and $n_1 \geq x_1$. The set of correct tuples of $\mathbb{N}^4$ is denoted $\mathbb{N}^4_{\mathsf{c}}$. For any r in Trans, we define the binary relation $\mathcal{R}_{r,\Delta}$ over the set $Q \times \mathbb{N}^4_{\mathsf{c}}$ as ($\mathcal{R}_{r,\Delta}$ is used in infix notation): $(q, (n_0, x_0, n_1, x_1)) \ \mathcal{R}_{r,\Delta} \ (q', (n_0', x_0', n_1', x_1'))$ if $(q, r, q') \in \Delta$ and for $j, k \in \{0, 1\}$, $j \neq k$, one has $n_k' = n_k$, $x_k' = x_k$ and one of the following holds: (i)

$r = \oplus_j$, $n'_j = n_j$, $x'_j = x_j - 1$, (ii) $r = \ominus_j$, $n'_j = n_j - 1$, $x'_j = x_j$ or (iii) $r = 0_j$, $n'_j = n_j = x_j = x'_j$.

Intuitively, the value of the first counter of $\mathcal{M}$ is obtained as $n_0 - x_0$ and the second one as $n_1 - x_1$. The binary relation $\mathcal{R}_\Delta$ is the union of the relations $\mathcal{R}_{r,\Delta}$ for all r in Trans and ${\mathcal{R}_\Delta}^*$ denotes its transitive and reflexive closure.

It holds that for any states q, q' of the machine and for any naturals n_0, n_1, n'_0, n'_1, $(q, n_0, n_1) \vdash^*_{\mathcal{M}} (q', n'_0, n'_1)$ iff there exist naturals x_0, x_1, x'_0, x'_1 such that $(q, (n_0 + x_0, x_0, n_1 + x_1, x_1))\ {\mathcal{R}_\Delta}^*\ (q', (n'_0 + x'_0, x'_0, n'_1 + x'_1, x'_1))$. Therefore, $\mathcal{L}(\mathcal{M})$ is not empty iff there exist some naturals $x, y, m_0, m'_0, m_1, m'_1$ such that $(q_i, (n_0 + x, x, y, y))\ {\mathcal{R}_\Delta}^*\ (q_f, (m_0, m'_0, m_1, m'_1))$.

Let $a_0[\{\!\}]$, $b_0[\{\!\}]$, $a_1[\{\!\}]$, $b_1[\{\!\}]$ be tree elements with a_0, b_0, a_1, b_1 four distinct labels. For any tuple of naturals (n_0, x_0, n_1, x_1) we denote $(\!|(n_0, x_0, n_1, x_1)|\!)$ the tree $n_0 \cdot a_0[\{\!\}] \uplus x_0 \cdot b_0[\{\!\}] \uplus n_1 \cdot a_1[\{\!\}] \uplus x_1 \cdot b_1[\{\!\}]$. For any $N \subseteq \mathbb{N}^4$, $(\!|N|\!)$ denotes the set of trees $\{(\!|\mathbf{n}|\!) \mid \mathbf{n} \in N\}$.

For any state $q \in Q$, we define $Acc(q)$ as

$$\{s \in \mathbb{N}^4_{\mathsf{c}} \mid \exists s' \in \mathbb{N}^4_{\mathsf{c}},\ (q, s)\ {\mathcal{R}_\Delta}^*\ (q_f, s')\}$$

We define a system of fixed point equations Σ over the set of variables $\{\xi_{\mathsf{ok}0}, \xi_{\mathsf{ok}1}, \xi_{\mathsf{ok}}, \xi_{\mathsf{zero}0}, \xi_{\mathsf{zero}1}, \xi\} \cup \{\xi_q \mid q \in Q\}$ such that $\mathsf{last}(\Sigma) = \xi$. This system will satisfy that for any $q \in Q$, $\mathcal{S}_\Sigma(\xi_q) = (\!|Acc(q)|\!)$. The first five equations of Σ are:

$$\xi_{\mathsf{ok}j} \stackrel{\mu}{=} a_j[\mathbf{0}] \,|\, b_j[\mathbf{0}] \,|\, \xi_{\mathsf{ok}j} \vee (a_j[\mathbf{0}])* \quad \text{for } j \text{ in } \{0,1\}$$

$$\xi_{\mathsf{ok}} \stackrel{\mu}{=} \xi_{\mathsf{ok}0} \,|\, \xi_{\mathsf{ok}1}$$

$$\xi_{\mathsf{zero}j} \stackrel{\mu}{=} a_j[\mathbf{0}] \,|\, b_j[\mathbf{0}] \,|\, \xi_{\mathsf{zero}j} \vee \xi_{\mathsf{ok}k} \quad \text{for } j, k \text{ in } \{0,1\}, k \neq j$$

These equations ensure that $\mathcal{S}_\Sigma(\xi_{\mathsf{ok}}) = \{(\!|s|\!) \mid s \in \mathbb{N}^4_{\mathsf{c}}\}$ and that $\mathcal{S}_\Sigma(\xi_{\mathsf{zero}j})$ is the set of trees $(\!|(n_0 + x_0, n_0, n_1 + x_1, n_1)|\!)$ with $n_0, n_1, x_0, x_1 \in \mathbb{N}$ and $x_j = 0$.

For any state $q \in Q$, $q \neq q_f$, the equation for q in Σ is

$$\xi_q \stackrel{\mu}{=} \bigvee_{(q,r,q') \in \Delta} \mathsf{Pred}(\xi_{q'}, r)$$

where for any r in Trans, $\mathsf{Pred}(\xi', r)$ is given by

$$\mathsf{Pred}(\xi', \oplus_j) = (\xi' \,|\, b_j[\mathbf{0}]) \wedge \xi_{\mathsf{ok}}$$
$$\mathsf{Pred}(\xi', \ominus_j) = (\xi' \,|\, a_j[\mathbf{0}]) \wedge \xi_{\mathsf{ok}}$$
$$\mathsf{Pred}(\xi', 0_j) = \xi' \wedge \xi_{\mathsf{zero}j}$$

The equation for ξ_{q_f} in Σ is $\xi_{q_f} \stackrel{\mu}{=} \xi_{\mathsf{ok}}$ and the last equation of Σ is $\xi \stackrel{\mu}{=} \xi_{q_i} \wedge \xi_{\mathsf{zero}1}$.

Intuitively, for a set $N \subseteq \mathbb{N}^4_{\mathsf{c}}$ of correct tuples and a valuation associating $(\!|N|\!)$ with ξ, $\mathsf{Pred}(\xi, r)$ represents the set of trees $(\!|s|\!)$ for s in $\mathbb{N}^4$ such that there exist a tuple $s' \in N$ and states $q, q' \in Q$ for which $(q, s)\ \mathcal{R}_{r,\Delta}\ (q', s')$

Thus, $\mathcal{L}(\mathcal{M})$ is empty iff $\mathcal{S}_\Sigma(\xi) = \varnothing$.

Corollary 7 *For any recursively enumerable set of naturals E, there exists a formula ϕ_E such that $n \in E$ iff there exist two naturals x, y such that the tree $(n + x) \cdot a_0[\{\!\}] \uplus x \cdot b_0[\{\!\}] \uplus y \cdot a_1[\{\!\}] \uplus y \cdot b_1[\{\!\}]$ belongs to $[\![\phi_E]\!]$, where a_0, b_0, a_1, b_1 are pairwise distinct labels.*

Proposition 8 *There exists a STL formula ϕ such that $[\![\phi]\!]$ is not PMSO definable.*

Sketch of proof By Proposition 4, it is enough to show that there exists a STL formula ϕ such that $[\![\phi]\!]$ can not be represented as the language of some semilinear constraint automaton. Let Λ' be the set of labels $\{a_0, b_0, a_1, b_1\}$. By Corollary 7, let ϕ_E be the formula associated with a recursively enumerable set of naturals E, *ie* s.t. $n \in E$ iff the tree $(n + x) \cdot a_0[\{\!\}] \uplus x \cdot b_0[\{\!\}] \uplus y \cdot a_1[\{\!\}] \uplus y \cdot b_1[\{\!\}]$ belongs to $[\![\phi_E]\!]$ for some naturals x, y. Consider now the automaton $A = (\Lambda, Q, \Delta, F)$ with $Q = \bigcup_{c \in \Lambda'} \{q_c\}$ and for any $c \in \Lambda'$, $\Delta(q_c) = (0^Q, \{c\})$. One can prove that $[\![\phi_E]\!]$ is equal to the language of some semilinear constrained automaton iff $[\![\phi_E]\!]$ is equal to $\mathcal{L}(A)$ for some semilinear set of mappings F. Assume that such a set F exists, then by definition of ϕ_E, there exist $x, y \in \mathbb{N}$ such that F is the set of mappings $\{(q_{a_0} \mapsto n + x, q_{b_0} \mapsto x, q_{a_1} \mapsto y, q_{b_1} \mapsto y) \mid n \in E\}$. Now, as semilinear sets are closed by projection and by linear combinations of components, the set of mappings $\{(q_{a_0} \mapsto n) \mid n \in E\}$ from $\mathbb{N}^{\{q_{a_0}\}}$ must be semilinear, implying that any recursively enumerable set of naturals is semilinear.

Theorem 9 *STL is strictly more expressive than the logic PMSO.*

Proof Follows from Proposition 8 and Proposition 25 from Section 6 stating that semilinear constrained automata are equally expressive to a fragment of STL. □

5 Tree automata for STL

All through this section we consider a closed STL formula ϕ. We will define an automaton A_ϕ such that $\mathcal{L}(A_\phi) = [\![\phi]\!]$.

A system of equations is in *normal form* if the right-hand side of each equation has one of the following forms:

$$\top \quad \mathbf{0} \quad \overline{\mathbf{0}} \quad \xi \vee \xi' \quad \xi \wedge \xi' \quad \alpha[\xi] \quad \overline{\alpha}[\xi] \quad \xi \,|\, \xi' \quad \xi \,\|\, \xi'$$

Equations of the form $\xi \stackrel{\kappa}{=} \alpha[\xi']$ or $\xi \stackrel{\kappa}{=} \overline{\alpha}[\xi']$ are called *elementary equations* and $\mathsf{EVars}(\Sigma)$ is the set of variables of Σ occurring as left-hand side of some elementary equation.

Any system Σ with variables $\{\xi_1, \ldots, \xi_n\}$ can be transformed into a system Σ' in normal form with some fresh extra variables $\{\xi'_1, \ldots, \xi'_m\}$; this transformation preserves the solution in the sense that for all $1 \in 1..n$, the solutions of the systems Σ and Σ' for the variable ξ_i coincide.

A system of equations Σ is *strongly normalised* if it is normalised and for any equation $\xi \overset{\kappa}{=} \xi' \,|\, \xi''$ or $\xi \overset{\kappa}{=} \xi' \,\|\, \xi''$ in Σ, $\{\!|\}\!|$ does not belong neither to $\mathcal{S}_\Sigma(\xi')$ nor to $\mathcal{S}_\Sigma(\xi'')$.

For any closed STL formula ϕ, the system of equations $\mathsf{Eq}(\phi)$ can be transformed into a strongly normalised system of equations denoted $\tilde{\mathsf{Eq}}(\phi)$: we first put $\mathsf{Eq}(\phi)$ in normal form, thus obtaining a system of equations Σ. Let $\xi \overset{\kappa}{=} \xi' \,|\, \xi''$ be an equation in Σ and let $\{\!|\}\!|$ belong to $\mathcal{S}_\Sigma(\xi'')$ but not to $\mathcal{S}_\Sigma(\xi')$. Then one can introduce a new variable to Σ, say ξ''_{n}, with corresponding equation $\xi''_{\mathsf{n}} \overset{\kappa}{=} \xi'' \wedge \overline{\mathbf{0}}$ and replace in Σ the equation for ξ by $\xi \overset{\kappa}{=} \xi' \mid \xi''_{\mathsf{n}} \vee \xi'$. The newly obtained system of equations Σ' is equivalent to $\mathsf{Eq}(\phi)$ over the original variables. Σ' can then be put in normal form to obtain $\tilde{\mathsf{Eq}}(\phi)$. This can effectively be done as:

Lemma 10 *For any system of equations Σ and for any variable ξ in $\mathsf{Vars}(\Sigma)$, one can decide whether $\{\!|\}\!|$ belongs to $\mathcal{S}_\Sigma(\xi)$.*

Example 2 *A system of equations Σ in normal form for the formula ϕ given in Example 1 is*

$$\xi'_1 \overset{\nu}{=} \xi_1 \,\|\, \xi_2, \quad \xi'_2 \overset{\nu}{=} \overline{\mathbf{0}}, \quad \xi_1 \overset{\nu}{=} \xi'_1 \wedge \xi'_2,$$
$$\xi'_3 \overset{\mu}{=} \alpha[\xi_1], \quad \xi'_4 \overset{\mu}{=} \xi'_3 \,|\, \xi_2, \quad \xi'_5 \overset{\mu}{=} \mathbf{0}, \quad \xi_2 \overset{\mu}{=} \xi'_4 \vee \xi'_5$$

It holds for Σ that $\{\!|\}\!|$ belongs to $\mathcal{S}_\Sigma(\xi_2)$ and $\mathcal{S}_\Sigma(\xi'_5)$ and does not belong to $\mathcal{S}_\Sigma(\xi)$ for any variable ξ in $\mathsf{Vars}(\Sigma)$ different from ξ_2 and ξ'_5. The system of equations Σ' is:

$$\xi'_1 \overset{\nu}{=} \xi_1 \,\|\, \xi''_2 \vee \xi_1, \quad \xi'_2 \overset{\nu}{=} \overline{\mathbf{0}}, \quad \xi_1 \overset{\nu}{=} \xi'_1 \wedge \xi'_2,$$
$$\xi'_3 \overset{\mu}{=} \alpha[\xi_1], \quad \xi'_4 \overset{\mu}{=} \xi'_3 \,|\, \xi''_2 \vee \xi'_3, \quad \xi''_5 \overset{\mu}{=} \mathbf{0} \wedge \overline{\mathbf{0}},$$
$$\xi'_5 \overset{\mu}{=} \mathbf{0}, \quad \xi''_2 \overset{\mu}{=} \xi_2 \wedge \overline{\mathbf{0}}, \quad \xi_2 \overset{\mu}{=} \xi'_4 \vee \xi'_5$$

$$A_\phi$$

States of the automaton For any $\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$, we define the STL formulae $\mathsf{Pos}(\xi)$, $\mathsf{Neg}(\xi)$ and $\mathsf{Compl}(\xi)$ depending on the equation for the variable ξ in $\tilde{\mathsf{Eq}}(\phi)$

$$\text{if } \xi \overset{\kappa}{=} \alpha[\xi'] \quad \begin{cases} \mathsf{Pos}(\xi) = \alpha[\xi'] \\ \mathsf{Neg}(\xi) = \alpha[\neg\xi'] \\ \mathsf{Compl}(\xi) = \overline{\alpha}[\top] \end{cases}$$
$$\text{if } \xi \overset{\kappa}{=} \overline{\alpha}[\xi'] \quad \begin{cases} \mathsf{Pos}(\xi) = \overline{\alpha}[\xi'] \\ \mathsf{Neg}(\xi) = \overline{\alpha}[\neg\xi'] \\ \mathsf{Compl}(\xi) = \alpha[\top] \end{cases}$$

We denote $\mathsf{split}(\xi)$ the set $\{\mathsf{Pos}(\xi), \mathsf{Neg}(\xi), \mathsf{Compl}(\xi)\}$.

The set of states Q_ϕ of the automaton A_ϕ is defined as $Q_\phi = \prod_{\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))} \mathsf{split}(\xi)$ where $\prod$ denotes the Cartesian product. For any $\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$, and for any state $q \in Q_\phi$, we denote with $q(\xi)$ the component of q corresponding to ξ.

Transitions and acceptance condition of the automaton We transform the system of equations $\tilde{\mathsf{Eq}}(\phi)$ into the system of equations $\mathsf{Eqn}(\phi)$ over the same set of variables obtained from $\tilde{\mathsf{Eq}}(\phi)$ by replacing each elementary equation $\xi \overset{\kappa}{=} \alpha[\xi']$ or $\xi \overset{\kappa}{=} \overline{\alpha}[\xi']$ by $\xi \overset{\kappa}{=} \mathsf{prj}_\xi$ (the prj_ξs are constants whose value will be given shortly.)

For two sets of mappings $I, I' \subseteq \mathbb{N}^Q$ (for some finite set Q), the sets of mappings $I + I'$ and $I +\!\!+ I'$ are defined by:

$$I + I' = \{\mathbf{d}'' \mid \exists \mathbf{d}, \mathbf{d}'.\, \mathbf{d}'' = \mathbf{d} + \mathbf{d}' \text{ and } \mathbf{d} \in I \text{ and } \mathbf{d}' \in I'\}$$
$$I +\!\!+ I' = \{\mathbf{d}'' \mid \forall \mathbf{d}, \mathbf{d}'.\, \mathbf{d}'' \neq \mathbf{d} + \mathbf{d}' \text{ or } \mathbf{d} \in I \text{ or } \mathbf{d}' \in I'\}$$

We interpret $\mathsf{Eqn}(\phi)$ on the complete lattice $(\wp(\mathbb{N}^{Q_\phi}), \vee, \wedge)$: Boolean operators $\vee, \wedge$ are interpreted as union and intersection; $\mathbf{0}$ is the singleton set $\{0^{Q_\phi}\}$, $\overline{\mathbf{0}}$ is the set of mappings $\mathbb{N}^{Q_\phi} \setminus \{0^{Q_\phi}\}$ and $\top$ is $\mathbb{N}^{Q_\phi}$; for any sets of mappings $I, I' \subseteq \mathbb{N}^Q$, the symbols $|, \|$ are interpreted as the operators $+$ and $+\!\!+$ respectively. Finally, for any variable ξ in $\mathsf{EVars}(\mathsf{Eqn}(\phi))$, prj_ξ is interpreted as the set of mappings $\{\mathbf{n} \mid \mathbf{n}(q) = 1 \text{ if } q(\xi) = \mathsf{Pos}(\xi) \text{ and } \mathbf{n}(q) = 0 \text{ otherwise}\}$. The solution of $\mathsf{Eqn}(\phi)$ over the lattice $(\wp(\mathbb{N}^{Q_\phi}), \vee, \wedge)$ for the variable ξ is denoted $\mathcal{R}_{\mathsf{Eqn}(\phi)}(\xi)$.

For any ξ in $\mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$, the set of labels $\mathsf{lab}(\xi)$ and the variable $\mathsf{var}(\xi)$ are given by: $\mathsf{lab}(\xi) = \alpha$ and $\mathsf{var}(\xi) = \xi'$ if $\xi \overset{\kappa}{=} \alpha[\xi']$ is an equation in $\tilde{\mathsf{Eq}}(\phi)$ and $\mathsf{lab}(\xi) = \Lambda \setminus \alpha$ and $\mathsf{var}(\xi) = \xi'$ if $\xi \overset{\kappa}{=} \overline{\alpha}[\xi']$ is an equation in $\tilde{\mathsf{Eq}}(\phi)$.

Now, for any state $q \in Q_\phi$, let α_q be the set of labels and N_q the set of mappings from $\mathbb{N}^{Q_\phi}$ defined as (where $\mathsf{PosV}(q) \subseteq \mathsf{Vars}(\mathsf{Eqn}(\phi))$ is the set of variables ξ for which $q(\xi) = \mathsf{Pos}(\xi)$ and the sets $\mathsf{NegV}(q)$ and $\mathsf{ComplV}(q)$ are defined accordingly):

$$\alpha_q = \bigcap_{\xi \in \mathsf{PosV}(q) \cup \mathsf{NegV}(q)} \mathsf{lab}(\xi) \cap \bigcap_{\xi \in \mathsf{ComplV}(q)} \Lambda \setminus \mathsf{lab}(\xi)$$
$$N_q^{\mathsf{Pos}} = \bigcap_{\xi \in \mathsf{PosV}(q)} \mathcal{R}_{\mathsf{Eqn}(\phi)}(\mathsf{var}(\xi))$$
$$N_q^{\mathsf{Neg}} = \bigcap_{\xi \in \mathsf{NegV}(q)} \mathbb{N}^{Q_\phi} \setminus \mathcal{R}_{\mathsf{Eqn}(\phi)}(\mathsf{var}(\xi))$$
$$N_q = N_q^{\mathsf{Pos}} \cup N_q^{\mathsf{Neg}}$$

Then for any $q \in Q_\phi$, $\Delta_\phi(q)$ is defined as (N_q, α_q).

Finally, F_ϕ, the acceptance condition of A_ϕ, is the set of mappings $\mathcal{R}_{\mathsf{Eqn}(\phi)}(\mathsf{last}(\mathsf{Eqn}(\phi)))$.

Note that the automaton A_ϕ is deterministic by construction.

We show in this section that the construction of the automaton we gave is correct, that is, $[\![\phi]\!] = \mathcal{L}(A_\phi)$. We start

by defining the notion of basis and give some of their properties needed for our proof.

5.3.1 Bases and their properties

Definition 11 (Bases) *Let Q be a finite set. A* basis Θ over Q *is a mapping in $\wp(\mathsf{ETree})^Q$ verifying that $\Theta(q) \cap \Theta(q') = \varnothing$ for any $q, q' \in Q, q \neq q'$ and $\bigcup_{q \in Q} \Theta(q) = \mathsf{ETree}$.*

Definition 12 (Bases product) *Let $Q_1, \ldots, Q_n$ be some finite sets. For n bases $\Theta_1, \ldots, \Theta_n$ over $Q_1, \ldots, Q_n$ respectively, their* product, *denoted $\prod_{i \in 1..n} \Theta_i$, is the mapping in $\mathbb{N}^{Q_1 \times \cdots \times Q_n}$ defined by $(\prod_{i \in 1..n} \Theta_i)((q_1, \ldots, q_n)) = \bigcap_{i \in 1..n} \Theta_i(q_i)$ for all $(q_1, \ldots, q_n) \in Q_1 \times \ldots \times Q_n$.*

Lemma 13 *Let $Q_1, \ldots, Q_n$ be some finite sets. If $\Theta_1, \ldots, \Theta_n$ are bases over $Q_1, \ldots, Q_n$ respectively, then the product $\prod_{i \in 1..n} \Theta_i$ is a basis over $Q_1 \times \cdots \times Q_n$.*

Definition 14 (Decomposition) *Let Q be a finite set and Θ be a basis over Q.*
• *For any tree $t = \{\!\{e_1, \ldots, e_n\}\!\}$, the* decomposition *of t on Θ, written $\mathsf{dcmp}(t, \Theta)$, is the mapping in $\mathbb{N}^Q$ associating with each $q \in Q$ the cardinality of the multiset $\Theta(q) \cap \{\!\{e_1, \ldots, e_n\}\!\}$.*
• *For any set of trees T, the* decomposition *of T on Θ, written $\mathsf{dcmp}(T, \Theta)$, is the set of mappings $\{\mathsf{dcmp}(t, \Theta) \mid t \in T\}$.*

In other words, $\mathsf{dcmp}(t, \Theta)(q)$ is the number of tree elements among $e_1, \ldots, e_n$ that belong to the set $\Theta(q)$.

Definition 15 (Interpretation) *Let Q be a finite set and Θ be a basis over Q. For any mapping $\mathbf{d} \in \mathbb{N}^Q$, the* interpretation *of $\mathbf{d}$ on Θ, written $[\![\mathbf{d}, \Theta]\!]$, is the set of trees t such that for all $q \in Q$, $(\mathsf{dcmp}(t, \Theta))(q) = \mathbf{d}(q)$ whenever $\Theta(q) \neq \varnothing$. Interpretation is extended to subsets of $\mathbb{N}^Q$ as follows: for any $D \subseteq \mathbb{N}^Q$, $[\![D, \Theta]\!] = \bigcup_{\mathbf{d} \in D} [\![\mathbf{d}, \Theta]\!]$.*

Note that the decomposition of a tree t on a basis Θ over Q is unique, whereas there may be several mappings $\mathbf{d} \in \mathbb{N}^Q$ such that $[\![\mathbf{d}, \Theta]\!] = t$. More precisely, for any q such that $\Theta(q) = \varnothing$, the value of $\mathbf{d}(q)$ does not contribute for $[\![\mathbf{d}, \Theta]\!]$ and thus may be arbitrary.

Definition 16 (Discrimination) *A basis Θ is* discriminating *for a set of trees T if $[\![\mathsf{dcmp}(T, \Theta), \Theta]\!] = T$.*

Informally, discrimination expresses that for any mapping $\mathbf{d}$ in D either all trees whose decomposition on Θ is $\mathbf{d}$ belong to T or no such tree belongs to T.

We terminate with some properties of bases and discrimination.

Proposition 17 *Let Q be a finite set and Θ a basis over Q.*

1. *Θ is discriminating for the set of trees T iff it is discriminating for $\mathsf{Tree} \setminus T$. Moreover in this case $[\![\mathbb{N}^Q \setminus \mathsf{dcmp}(T, \Theta), \Theta]\!] = \mathsf{Tree} \setminus T$.*
2. *$\mathsf{Tree} = [\![\mathbb{N}^Q, \Theta]\!]$ and $\{\!\{\}\!\} = [\![0^Q, \Theta]\!]$.*
3. *For any sets of trees T, T' s.t. Θ is discriminating for T and T', Θ is discriminating for the sets of trees $T \cup T'$, $T \cap T'$, $T \,|\, T'$ and $T \,\|\, T'$. Moreover, if $T = [\![D, \Theta]\!]$ and $T' = [\![D', \Theta]\!]$, then $T \vee T' = [\![D \vee D', \Theta]\!]$, $T \wedge T' = [\![D \wedge D', \Theta]\!]$, $T \mid T' = [\![D + D', \Theta]\!]$ and $T \parallel T' = [\![D +\!\!+ D', \Theta]\!]$.*

5.3.2 Proof of correctness

Now, with any ξ in $\mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$ we associate the mapping $\mathsf{basis}(\xi)$ in $\wp(\mathsf{ETree})^{\mathsf{split}(\xi)}$ defined as:

$$\mathsf{basis}(\xi)(\mathsf{Pos}(\xi)) = \mathsf{lab}(\xi)[\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\mathsf{var}(\xi))]$$
$$\mathsf{basis}(\xi)(\mathsf{Neg}(\xi)) = \mathsf{lab}(\xi)[\mathsf{Tree} \setminus \mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\mathsf{var}(\xi))]$$
$$\mathsf{basis}(\xi)(\mathsf{Compl}(\xi)) = (\Lambda \setminus \{\mathsf{lab}(\xi)\})[\mathsf{Tree}]$$

It is easy to see that $\mathsf{basis}(\xi)$ is a basis over $\mathsf{split}(\xi)$.

Consider now the basis Θ_ϕ defined as the product of the bases $\mathsf{basis}(\xi)$ for any $\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$, that is, $\Theta_\phi = \prod_{\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))} \mathsf{basis}(\xi)$. Remind now that the set Q_ϕ is defined as $\prod_{\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))} \mathsf{split}(\xi)$; so, by definition of $\mathsf{basis}(\xi)$ and bases product, we conclude that Θ_ϕ is a basis over Q_ϕ. We will show that for any $q \in Q_\phi$, $\mathcal{L}(q) = \Theta_\phi(q)$. This proof requires auxiliary results we give first.

Proposition 18 *For any ξ variable in $\tilde{\mathsf{Eq}}(\phi)$, Θ_ϕ is discriminating for $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi)$.*

Sketch of proof Towards contradiction: let $\mathbf{d}$ be a minimal (for a well-founded partial ordering over mappings) mapping and ξ be a variable in $\mathsf{Vars}(\tilde{\mathsf{Eq}}(\phi))$ such that there exist trees t, t' for which $\mathsf{dcmp}(t, \Theta_\phi) = \mathsf{dcmp}(t', \Theta_\phi) = \mathbf{d}$ and $t \in \mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi)$ and $t' \notin \mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi)$ (that is, Θ_ϕ is not discriminating for $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi)$). One can show that the right-hand side of this equation is of the form $\xi' \mid \xi''$ or $\xi' \parallel \xi''$. As $\tilde{\mathsf{Eq}}(\phi)$ is strongly normalised system of equations and so $\{\!\{\}\!\}$ does not belong to $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi')$ neither to $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi'')$, there exists a mapping $\mathbf{d}'$ strictly smaller than $\mathbf{d}$, a variable $\xi_1 \in \mathsf{Vars}(\tilde{\mathsf{Eq}}(\phi))$ and trees t_1, t_2 such that $\mathsf{dcmp}(t_1, \Theta_\phi) = \mathsf{dcmp}(t_2, \Theta_\phi) = \mathbf{d}'$ and $t_1 \in \mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi_1)$ and $t_2 \notin \mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi_1)$. This contradicts the minimality of $\mathbf{d}$ and allows to conclude that Θ_ϕ is discriminating for $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi)$ for any variable ξ in $\mathsf{Vars}(\tilde{\mathsf{Eq}}(\phi))$.

Proposition 19 *For any ξ variable in $\tilde{\mathsf{Eq}}(\phi)$, $\mathcal{S}_{\tilde{\mathsf{Eq}}(\phi)}(\xi) = [\![\mathcal{R}_{\mathsf{Eqn}(\phi)}(\xi), \Theta_\phi]\!]$.*

Sketch of proof For any $\xi \in \mathsf{EVars}(\tilde{\mathsf{Eq}}(\phi))$, one has $\mathcal{S}(\xi) = [\![\mathsf{prj}_\xi, \Theta_\phi]\!]$ by definition of Θ_ϕ and prj_ξ; so the property holds in this case. For the other variables, using Proposition 17 and definition of the system of equations $\tilde{\mathsf{Eq}}(\phi)$ we can conclude that the solution of $\tilde{\mathsf{Eq}}(\phi)$ over the lattice $(\wp(\mathsf{Tree}), \vee, \wedge)$ is exactly the solution of $\tilde{\mathsf{Eq}}(\phi)$ over the lattice $(\wp(\mathbb{N}^{\Theta_\phi}), \vee, \wedge)$ but using an alternative representation of sets of trees as their decomposition on the basis Θ_ϕ.

Proposition 20 *For any state q in Q_ϕ, $\mathcal{L}(q) = \Theta_\phi(q)$. For any set of mappings N included in $\mathbb{N}^{Q_\phi}$, $\mathcal{L}(N) = [\![N, \Theta_\phi]\!]$.*

Sketch of proof The proof goes by induction on the height of trees; denoting Tree_n, (resp. ETree_n) the set of trees (resp. of tree elements) of height at most n, we show that for any natural n, $\mathcal{L}(q) \cap \mathsf{ETree}_n = \Theta_\phi(q) \cap \mathsf{ETree}_n$ and $\mathcal{L}(N) \cap \mathsf{Tree}_n = [\![N, \Theta_\phi]\!] \cap \mathsf{Tree}_\phi$.

Theorem 21 $[\![\phi]\!] = \mathcal{L}(A_\phi)$.

Proof From Proposition 20 as $\mathcal{L}(A_\phi) = \mathcal{L}(F_\phi)$. □

Using Proposition 2 and the construction of A_ϕ, one can give sufficient conditions on the system of equations $\mathsf{Eqn}(\phi)$ that guarantee decidability of emptiness for A_ϕ.

Proposition 22 *If for any two disjoint sets $W, W' \subseteq \mathsf{Vars}(\mathsf{Eqn}(\phi))$ and for $Q' \subseteq Q_\phi$, one can decide whether there exists $\mathbf{n}$ in $\bigcap_{\xi \in W} \mathcal{R}_{\mathsf{Eqn}(\phi)}(\xi) \cap \bigcap_{\xi \in W'} \mathbb{N}^{Q_\phi} \setminus \mathcal{R}_{\mathsf{Eqn}(\phi)}(\xi)$ such that $\mathbf{n}(q) \neq 0$ implies $q \in Q'$, then emptiness is decidable for the automaton A_ϕ.*

6 Fragments of STL with limited recursion

We define here two syntactic fragments of STL, namely gSTL and pgSTL; we show that the former is equivalent to star-free constrained automata and the latter to semilinear constrained automata. This implies their equivalence with MSO and PMSO respectively.

We assume in this section that STL formulae are given using the initial syntax of the logic (no derived operators) but allowing the star operator. An occurrence of a variable ξ is *guarded* in some closed formula ϕ if it appears under some nesting operator $\alpha[\ ]$ in the sub-formula $\mu\xi.\phi'$ of ϕ. A STL formula ϕ is said to be *guarded* (*for recursion*), if all occurrences of recursion variables are guarded in ϕ. A STL formula is said to be *positively guarded* (*for recursion*), if for any $\mu\xi.\phi'$ sub-formula of ϕ, any occurrence of the recursion variable ξ in ϕ' is either guarded or not preceded by any negation in ϕ'.

Definition 23 (gSTL and pgSTL logics) *The logic gSTL (resp. pgSTL) is the restriction of STL to guarded formulae (resp. to positively guarded formulae) for recursion.*

Note that the gSTL fragment is included in the pgSTL fragment. Remark also that the star operator can be used in pgSTL (as it can be defined in it) but not in gSTL.

Example 3 *Let $\phi_1 = \alpha[\beta[\xi] \mid \neg(\neg\xi \mid \neg c[\mathbf{0}])] \vee \mathbf{0}$, $\phi_2 = \mu\xi.\alpha[\xi]* \mid (\beta[\mathbf{0}] \vee \xi) \vee \mathbf{0}$ and $\phi_3 = \mu\xi.\neg(\beta[\mathbf{0}] \mid \neg(\alpha[\mathbf{0}] \mid \xi)) \vee \mathbf{0}$. The formula ϕ_1 is a gSTL formula, as all occurrences of ξ are guarded by the nesting operator $\alpha[]$; the formula ϕ_2 is a pgSTL formula, as the first occurrence of ξ is guarded by the nesting operator α and the second one is not preceded by any negation; the formula ϕ_3 is neither a gSTL formula nor a pgSTL formula, as the third occurrence of the variable ξ is not guarded and is preceded by negations.*

Intuitively, gSTL restricts recursion to vertical traversing of trees, whereas pgSTL allows vertical and (a restricted form of) horizontal recursion. It has to be noticed that most reasonable properties on trees can be expressed in pgSTL.

We establish the equivalences between the gSTL fragment of the logic and star-free constrained automata on one hand and the pgSTL fragment and semilinear constrained automata on the other hand.

Proposition 24 *If ϕ is a formula from gSTL (resp. from pgSTL), then for any variable ξ from $\tilde{\mathsf{Eq}}(\phi)$, $\mathcal{R}_{\mathsf{Eqn}(\phi)}(\xi)$ is a star-free (resp. a semilinear) set of mappings that can be given effectively.*

Theorem 25 *For any set of trees T, T is gSTL definable iff it is accepted by some star-free constrained automata.*

For any set of trees T, T is pgSTL definable iff it is accepted by some semilinear constrained automaton.

Corollary 26 *Satisfiability is decidable for the logics gSTL and pgSTL.*

Proof Testing satisfiability for a formula ϕ reduces to testing emptiness for the corresponding automaton A_ϕ. Conclusion follows from Proposition 24 and Corollary 3. □

Moreover, it can be shown that the pgSTL fragment of the logic collapses in the fragment of guarded and negation-free STL formulae enriched with the star operator.

From the encoding of two-counter machines used in Theorem 6 to show undecidability of the full STL and from decidability result for pgSTL, it seems that the source of undecidability of the full logic is the possibility of mixing horizontal recursion and conjunction in full STL.

Theorem 27 *For any set of trees T, T is MSO definable iff T is gSTL definable and T is PMSO definable iff T is pgSTL definable.*

7 Final Remarks

From any pgSTL formula, we have shown that an equivalent PMSO formula can be built. Therefore, using results on model-checking from [20], we have that the data complexity for the model-checking problem for pgSTL is linear in the size of the tree. Using our tree automata approach, one can show that data complexity of model checking for the full STL logic is polynomial time.

Decidability results for the STL logic presented in this paper still hold when the set of labels Λ is countable, as it is the case for the logic TL presented in [12, 13].

We strongly believe that the satisfiability for pgSTL can be decided in elementary time. This would imply that the construction of a pgSTL formula from a given PMSO formula is non-elementary.

Acknowledgements The authors thank G. Ghelli and D. Colazzo for their valuable comments on a preliminary version of this work.

References

[1] A. Arnold and D. Wink. *Rudiments of μ-Calculus*. North-Holland, 2001.

[2] I. Boneva and J.-M. Talbot. On Complexity of Model-Checking for the TQL Logic. In *3rd IFIP Int. Conf. on Theoretical Computer Science (TCS2004)*, pages 381–394, 2004.

[3] I. Boneva and J-M. Talbot. Automata and Logics for Unranked and Unordered Trees. In *16th Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 3467 of *LNCS*, pages 500–515, Springer, 2005.

[4] L. Caires and L. Cardelli. A Spatial Logic for Concurrency (part I). *Information and Computation*, 186(2):194–235, 2003.

[5] C. Calcagno, L. Cardelli, and A. D. Gordon. Deciding Validity in Spatial Logic for Trees. In *ACM SIGPLAN int. workshop on Types in languages design and implementation*, pages 62–73, 2003.

[6] L. Cardelli, P. Gardner, and G. Ghelli. A Spatial Logic for Querying Graphs. In *29th Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380 of *LNCS*, pages 597–610. Springer, 2002.

[7] L. Cardelli and G. Ghelli. A Query Language Based on the Ambient Logic. In *European Symp. on Programming (ESOP)*, volume 2028 of *LNCS*, pages 1–22. Springer, 2001.

[8] L. Cardelli and G. Ghelli. TQL: A Query Language for Semistructured Data Based on the Ambient Logic. *Mathematical Structures in Computer Science*, 14:285–327, 2004.

[9] L. Cardelli and A. D. Gordon. Anytime, Anywhere: Modal Logics for Mobile Ambients. In *27th ACM Symp. on Principles of Programming Languages (POPL'00)*, pages 365–377. ACM, 2000.

[10] L. Cardelli and A. D. Gordon. Mobile Ambients. *TCS*, 240:177–213, 2000.

[11] W. Charatonik, S. Dal Zilio, A. D. Gordon, S. Mukhopadhyayd, and J.-M. Talbot. Model Checking Mobile Ambients. *TCS*, 308(3):277–331, 2003.

[12] S. Dal Zilio and D. Lugiez. XML Schema, Tree Logic and Sheaves Automata. In *Rewriting Techniques and Applications, 14th Int. Conf. (RTA)*, volume 2706 of *LNCS*, pages 246–263. Springer, 2003.

[13] S. Dal Zilio, D. Lugiez, and C. Meyssonnier. A Logic You Can Count on. In *31st ACM SIGPLAN-SIGACT symp. on Principles of programming languages*, pages 135–146. ACM, 2004.

[14] A. Dawar, P. Gardner, and G. Ghelli. Expressiveness and Complexity of Graph Logic. Technical report, Imperial College, 2004.

[15] J. Goubault-Larrecq and K. N. Verma. Alternating Two-way AC-Tree Automata. Research Report, Laboratoire Spécification et Vérification, November 2002.

[16] D. Lugiez. Counting and Equality Constraints for Multitree Automata. In *Foundations of Software Science and Computational Structures: 6th Int. Conf., FOSSACS*, volume 2620 of *LNCS*, pages 328 – 342. Springer, 2003.

[17] M. L. Minsky. Recursive insolvability of Post's problem of "tag" ard other topics in the theory of turing machines. In *Annals of Mathematics, Second Series*, volume 74, pages 437–455, 1961.

[18] P. W. O'Hearn, J. C. Reynolds, and H. Yang:. Local Reasoning about Programs that Alter Data Structures. In *Computer Science Logic (CSL)*, volume 2142 of LNCS, pages 1–19. Springer, 2001.

[19] J C. Reynolds. Separation Logic: A Logic for Shared Mutable Data Structures. In *17th IEEE Symp. on Logic in Computer Science (LICS)*, pages 55–74. IEEE Computer Society, 2002.

[20] H. Seidl, T. Schwentick, and A. Muscholl. Numerical Document Queries. In *22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, pages 155–166. ACM, 2003.