EPiC Series in Computer Science

Volume 36, 2015, Pages 229–251

GCAI 2015. Global Conference on Artificial Intelligence

A Uniform Approach to Incremental Automated Reasoning on Strongly Distributed Structures

Elena V. Ravve[1], Zeev Volkovich[1], and Gerhard W. Weber[2]

[1] Braude College,
Karmiel, Israel
{cselena,vlvolkov}@braude.ac.il
[2] Middle East Technical University,
Ankara, Turkey
gweber@metu.edu.tr

Abstract

We introduce the notion of strongly distributed structures and present a uniform approach to incremental automated reasoning on them. The approach is based on a systematic use of two logical reduction techniques: Feferman-Vaught reductions and syntactically defined translation schemes. The distributed systems are presented as logical structures $\mathcal{A}$'s. We propose a uniform template for methods, which allow for a certain cost evaluation of formulae of logic $\mathcal{L}$ over $\mathcal{A}$ from values of formulae over its components and values of formulae over the index structure $\mathcal{I}$. Given logic $\mathcal{L}$, structure $\mathcal{A}$ as a composition of structures $\mathcal{A}_i, i \in I$, index structure $\mathcal{I}$ and formula ϕ of the logic to be evaluated on $\mathcal{A}$, the question is: what is the reduction sequence for ϕ if any. We show that if we may prove preservation theorems for $\mathcal{L}$ as well as if $\mathcal{A}$ is a strongly distributed composition of its components then the corresponding reduction sequence for $\mathcal{A}$ may be effectively computed. We show that the approach works for many extensions of FOL but not for all. The considered extensions of FOL are suitable candidates for modeling languages for components and services, used in incremental automated reasoning, data mining, decision making, planning and scheduling. A short complexity analysis of the method is also provided.

1 Introduction

Every day, more than 2 quintillion bytes of data are created and and 90% of the data in the world today was created within the past two years, cf. [20]. The maintenance and proceeding of such amount of data requires *"massively parallel software running on tens, hundreds, or even thousands of servers"*, cf. [23]. Incremental automated reasoning, data mining, decision making, planning and scheduling are hardly implemented on distributed systems. There are two schools of thought on reasoning about distributed systems: one following interleaving based semantics, and one following partial-order (or graph) based semantics, cf. [4]. The second one seems to be more promising. One of the successful attempts to specify the distributed multi-agent reasoning system ($dMARS$) is presented in [7]. However, to our best knowledge, no general, logically based approach to incremental reasoning has been proposed.

G.Gottlob, G.Sutcliffe and A.Voronkov (eds.), GCAI 2015 (EPiC Series in Computer Science, vol. 36), pp. 229–251

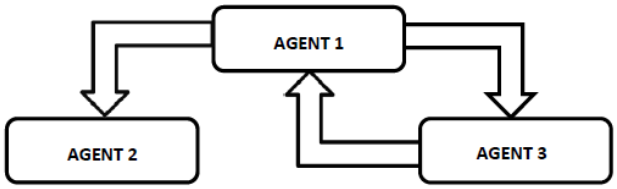

Figure 1: A system with 3 agents.

In this contribution, we systematically apply two logical reduction techniques to the field of reasoning on distributed systems. The distributed systems are presented as logical structures $\mathcal{A}$. We propose a uniform template for methods, which allow for a certain cost evaluation of formulae of logic $\mathcal{L}$ over $\mathcal{A}$ from values of formulae over its components and values of formulae over the index structure $\mathcal{I}$. The logical reduction techniques are:

1. **Feferman-Vaught reduction sequences** (or simply, reductions) were introduced in [14]. Given structure $\mathcal{A}$ as a composition of structures $\mathcal{A}_i, i \in I$, and index structure $\mathcal{I}$. A reduction sequence is a set of formulae such that each such a formula can be evaluated locally in some site or index set. Next, from the local answers, received from the sites, and possibly some additional information about the sites, we compute the result (boolean or even quantitative) for the given global formula. In the logical context, the reductions are applied to a relational structure $\mathcal{A}$ distributed over different sites with structures $\mathcal{A}_i, i \in I$. The reductions allow the formulae over $\mathcal{A}$ to be computed from formulae over the structures $\mathcal{A}_i$'s and formulae over index structure $\mathcal{I}$.
2. **Translation schemes** are the logical analogue to coordinate transformations in geometry. The fundamental property of translation schemes describes how to compute transformed formulae in the same way Leibniz' Theorem describes how to compute transformed integrals. The fundamental property has a long history, but was first properly stated by Rabin, cf. [38].

The paper is organized as follows. One of two already published motivating examples is presented in Section 2, the second is moved to Appendices A. In Section 3, we discuss different ways of obtaining structures from components. Section 4 introduces the notion of abstract translation schemes. Section 5 is the main section of the paper, where we state and prove our main Theorem 7. Short complexity analysis is presented in Section 6. Section 7 summarizes the paper and gives an outlook to future studies. Appendices B and C provide the logical notations, used in the contribution.

2 Motivating Example: Cooperation of Three Agents

This example is taken verbatim form [41] with few cosmetic changes. Assume we are given a system with three agents, which may communicate according to predefined rules. Assume that the **AGENT1** may call both agents: the **AGENT2** and the **AGENT3**, while the **AGENT2** can not call any agent and the **AGENT3** may call back the **AGENT1**; see Figure 1. We use the following formalization: the runs of the agents are presented as weighted labeled trees: the weights are put on the edges of the trees and the vertices may be labeled. Assume that a run tree T of the system is presented on Figure 2. In the figure, we omit the weights, put on the edges. The meaning of the labels on the vertices is as follows:

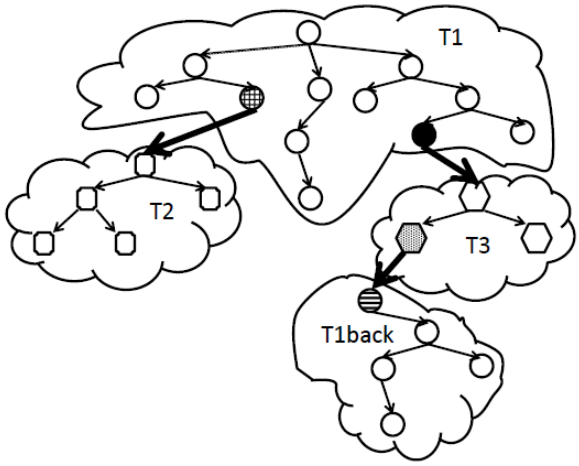

Figure 2: Run tree T of a system with 3 agents.

- the gridded vertex in $T1$ corresponds to the call to the **AGENT2** by the **AGENT1** (the bold edge goes to the root of $T2$);
- the filled vertex in $T1$ corresponds to the call to the **AGENT3** by the **AGENT1** (the bold edge goes to the root of $T3$);
- the doted vertex in $T3$ corresponds to the call to the **AGENT1** by the **AGENT3** (the bold edge does **NOT** go to the root of $T1$ but rather goes back to a vertex of $T1$ labeled by strips).

Assume that we want to optimize (minimize) the runs in the tree. On the one hand, we may use one of the optimization algorithm on the complete tree T that will give a quantitative result $\Re$. On the other hand, we observe that we may receive the optimal result in the following way:

1. We find the optimal run $\Re_1$ in $T1$.
2. We find **ALL** labeled runs Λ_{i_1} in $T1$: there are two such runs Λ_{1_1} and Λ_{2_1}.
3. We find the optimal run $\Re_2$ in $T2$. For the optimization, we will use $\Lambda_{1_1} + \Re_2$.
4. We find the optimal run $\Re_3$ in $T3$. For the optimization, we will use $\Lambda_{2_1} + \Re_3$.
5. We find **ALL** labeled runs Λ_{i_3} in $T3$: there is one such a run Λ_{1_3}.
6. We find the optimal run $\Re_{1_{back}}$ in $T1back$. We will use $\Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}$.
7. Finally, we find $\min\{\Re_1, \Lambda_{1_1} + \Re_2, \Lambda_{2_1} + \Re_3, \Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}\}$.
8. We observe that $\Re = \min\{\Re_1, \Lambda_{1_1} + \Re_2, \Lambda_{2_1} + \Re_3, \Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}\}$.

In order to generalize the obtained observations of the example, we need:

1. The precise definition of the weighted labeled trees and computations on them as well as the formal definition of a language that describes our optimization problems: Section C.
2. Tree T is **NOT** a disjoint union of its sub-trees. We need a formal framework to deal with such objects: Section 5.
3. Let $\Im_{old}(N)$ denote the time to solve the problem directly (N denotes the size of the coding of T).
   - $\mathcal{E}_{\mathcal{I}}$ denotes time to extract index structure I from T. We have four ordered numbers to distinguish sub-trees in our example.

- $\mathcal{E}_\imath$ denotes time to extract each $T_\imath$ from T. We have four numbered sub-trees in our example.
- $\mathcal{C}_\imath(n_\imath)$ denotes time to compute values $\Re_1$, Λ_{1_1}, Λ_{2_1} on $T1$; $\Re_2$ on $T2$; $\Re_3$, Λ_{1_3} on $T3$ and $\Re_{1_{back}}$ on $T1back$ ($n_\imath$ is the size of the coding of $T_\imath$).
- $\Im_F$ denotes time to build the sentence like $\min\{\Re_1, \Lambda_{1_1} + \Re_2, \Lambda_{2_1} + \Re_3, \Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}\}$.
- $\Im_{comp}$ denotes time to compute $\min\{\Re_1, \Lambda_{1_1} + \Re_2, \Lambda_{2_1} + \Re_3, \Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}\}$.

The new computation time is: $\Im_{new} = \mathcal{E}_{\mathcal{I}} + \Sigma_{\imath\in I}\mathcal{E}_\imath + \Sigma_{\imath\in I}\mathcal{C}_\imath + \Im_F + \Im_{comp}$. The question now is: When does hold $\Im_{old} > \Im_{new}$?

Moreover, we want the construction of our sentence like $\min\{\Re_1, \Lambda_{1_1} + \Re_2, \Lambda_{2_1} + \Re_3, \Lambda_{2_1} + \Lambda_{1_3} + \Re_{1_{back}}\}$ depends only upon the property to be optimized and the predefined "communication" rules, but **NOT** upon the given T.

3 Disjoint Union and Shuffling of Structures

The first reduction technique that we use is *Feferman-Vaught reductions*. Feferman-Vaught reduction sequence (or simply, reduction) is a set of formulae such that each such a formula can be evaluated locally in some component or index structure. Next, from the local values, received from the components, and possibly some additional information about the components, we compute the value for the given global formula. In the logical context, the reductions are applied to a relational structure $\mathcal{A}$ distributed over different components with structures $\mathcal{A}_i, i \in I$. The reductions allow the formulae over $\mathcal{A}$ can be computed from formulae over the structures $\mathcal{A}_i$'s and formulae over the index structure $\mathcal{I}$.

In this section, we start to discuss different ways of obtaining structures from components. The *Disjoint Union* of a family of structures is the simplest example of juxtaposing structures over an index structure $\mathcal{I}$ with universe I, where none of the components are linked to each other.

Definition 1 (Disjoint Union)**.**
Let $\tau_i = \langle R^i_1, \ldots, R^i_{j^i}\rangle$ be a vocabulary of structure $\mathcal{A}_i$. In the general case, the resulting structure is $\mathcal{A} = \dot{\bigsqcup}_{i\in I}\mathcal{A}_i = \langle I \cup \dot{\bigcup}_{i\in I}A_i, P(\imath, v), Index(x), R^I_j(1 \le j \le j^I), R^i_{j^i}(i \in I, 1 \le j^i \le j^\imath), \rangle$ for all $i \in I$, where $P(i, v)$ is true iff element a came from A_i, $Index(x)$ is true iff x came from I.

Definition 2 (Partitioned Index Structure)**.**
Let $\mathcal{I}$ be an index structure over τ_{ind}. $\mathcal{I}$ is called finitely partitioned *into ℓ parts if there are unary predicates I_α, $\alpha < \ell$, in the vocabulary τ_{ind} of $\mathcal{I}$ such that their interpretation forms a partition of the universe of $\mathcal{I}$.*

The following holds:

Theorem 1.
Let $\mathcal{I}$ be a finitely partitioned index structure. Let $\mathcal{A} = \dot{\bigsqcup}_{i\in I}\mathcal{A}_i$ be a τ–structure, where each $\mathcal{A}_i$ is isomorphic to some $\mathcal{B}_1, \ldots, \mathcal{B}_\ell$ over the vocabularies $\tau_1, \ldots, \tau_\ell$, in accordance to the partition (ℓ is the number of the classes). For every $\phi \in MSOL(\tau)$ there are:

- *a boolean function $F_\phi(b_{1,1}, \ldots, b_{1,j_1}, \ldots, b_{\ell,1}, \ldots, b_{\ell,j_\ell}, b_{I,1}, \ldots, b_{I,j_I})$*
- *$MSOL$–formulae $\psi_{1,1}, \ldots, \psi_{1,j_1}, \ldots, \psi_{\ell,1}, \ldots, \psi_{\ell,j_\ell}$*
- *$MSOL$–formulae $\psi_{I,1}, \ldots, \psi_{I,j_I}$*

such that for every $\mathcal{A}$, $\mathcal{I}$ and $\mathcal{B}_i$ as above with $\mathcal{B}_i \models \psi_{i,j}$ iff $b_{i,j} = 1$ and $\mathcal{B}_I \models \psi_{I,j}$ iff $b_{I,j} = 1$ we have

$$\mathcal{A} \models \phi \text{ iff } F_\phi(b_{1,1}, \ldots, b_{1,j_1}, \ldots, b_{\ell,1}, \ldots, b_{\ell,j_\ell}, b_{I,1}, \ldots, b_{I,j_I}) = 1.$$

Moreover, F_ϕ and the $\psi_{i,j}$ are computable from ϕ, ℓ and vocabularies alone, but are exponential in the quantifier rank of ϕ.

Proof: The proof is classical; see, in particular [3].

Now, we introduce an abstract preservation property of XX-combination of logics $\mathcal{L}_1, \mathcal{L}_2$, denoted by $XX - PP(\mathcal{L}_1, \mathcal{L}_2)$. XX may mean, for example, Disjoint Union. The reason why we look at this abstract property is that it can be proven for various logics using their associated pebble games. The proofs usually depend upon the details of the particular pebble games. However, the property $XX - PP(\mathcal{L}_1, \mathcal{L}_2)$ and its variants play an important role in our development of the Feferman-Vaught style theorems.

Definition 3 (Preservation Property with Fixed Index Set)**.**
For two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ we define Preservation Property for Disjoint Union

Input of operation: *Indexed set of structures;*

Preservation Property: *if for each $i \in I$ being the index set, $\mathcal{A}_i$, $\mathcal{B}_i$ satisfy the same sentences of $\mathcal{L}_1$, then the disjoint unions $\bigsqcup_{i\in I} \mathcal{A}_i$, $\bigsqcup_{i\in I} \mathcal{B}_i$ satisfy the same sentences of $\mathcal{L}_2$.*

Notation: $DJ - PP(\mathcal{L}_1, \mathcal{L}_2)$

The *Disjoint Union* of a family of structures is the simplest example of juxtaposing structures where none of the components are linked to each other. Another way of producing a new structure from several given structures is by mixing (shuffling) structures according to a (definable) prescribed way along the index structure.

Definition 4 (Shuffle over Partitioned Index Structure)**.**
Let $\mathcal{I}$ be a partitioned index structure into β parts, using unary predicates $I_\alpha, \alpha < \beta$. Let $\mathcal{A}_i, i \in I$, be a family of structures such that for each $i \in I_\alpha$ it holds $\mathcal{A}_i \cong \mathcal{B}_\alpha$, according to the partition. In this case, we say that $\bigsqcup_{i\in I} \mathcal{A}_i$ is the shuffle of $\mathcal{B}_\alpha$ along the partitioned index structure $\mathcal{I}$*, and denote it by $\biguplus^{\mathcal{I}}_{\alpha<\beta} \mathcal{B}_\alpha$.*

Note that the shuffle operation, as defined here, is a special case of the disjoint union, and that the disjoint pair is a special case of the finite shuffle.

Definition 5 (Preservation Properties with Variable Index Structures)**.**
For two logics $\mathcal{L}_1$ and $\mathcal{L}_2$ we define Preservation Properties for Shuffle

Input of operation: *A family of structures $\mathcal{B}_\alpha, \alpha < \beta$, and a (finitely) partitioned index structure $\mathcal{I}$ with I_α a partition.*

Preservation Property: *Assume that for each $\alpha < \beta$ the pair of structures $\mathcal{A}_\alpha, \mathcal{B}_\alpha$ satisfy the same sentences of $\mathcal{L}_1$, and $\mathcal{I}, \mathcal{J}$ satisfy the same $MSOL$-sentences. Then, the schuffles $\biguplus^{\mathcal{I}}_{\alpha<\beta} \mathcal{A}_\alpha$ and $\biguplus^{\mathcal{J}}_{\alpha<\beta} \mathcal{B}_\alpha$ satisfy the same sentences of $\mathcal{L}_2$.*

Notation: $Shu - PP(\mathcal{L}_1, \mathcal{L}_2)$ *($FShu - PP(\mathcal{L}_1, \mathcal{L}_2)$)*

Definition 6 (Reduction Sequence for Shuffling)**.**
Let $\mathcal{I}$ be a finitely partitioned τ_{ind}-index structure and $\mathcal{L}$ be logic.
Let $\mathcal{A} = \biguplus^{\mathcal{I}}_{\alpha<\beta} \mathcal{B}_\alpha$ be the τ–structure which is the finite shuffle of the τ_α-structures $\mathcal{B}_\alpha$ over $\mathcal{I}$.
A $\mathcal{L}_1$-reduction sequence for shuffling *for $\phi \in \mathcal{L}_2(\tau_{shuffle})$ is given by*

1. *a boolean function* $F_\phi(b_{1,1},\dots,b_{1,j_1},\dots,b_{\beta,1},\dots,b_{\beta,j_\beta},b_{I,1},\dots,b_{I,j_I})$
2. *set* Υ *of* $\mathcal{L}_1$*–formulae* $\Upsilon=\{\psi_{1,1},\dots,\psi_{1,j_1},\dots,\psi_{\beta,1},\dots,\psi_{\beta,j_\beta}\}$
3. $MSOL$*–formulae* $\psi_{I,1},\dots,\psi_{I,j_I}$

and has the property that for every $\mathcal{A}$, $\mathcal{I}$ *and* $\mathcal{B}_\alpha$ *as above with* $\mathcal{B}_\alpha\models\psi_{\alpha,j}$ *iff* $b_{\alpha,j}=1$, *and* $\mathcal{B}_I\models\psi_{I,j}$ *iff* $b_{I,j}=1$, *we have*

$$\mathcal{A}\models\phi \text{ iff } F_\phi(b_{1,1},\dots,b_{1,j_1},\dots,b_{\beta,1},\dots,b_{\beta,j_\beta},b_{I,1},\dots,b_{I,j_I})=1.$$

Note that we require that F_ϕ *and the formulae* $\psi_{\alpha,j}$ *depend only on* ϕ, β *and* $\tau_1,\dots,\tau_\beta$, *but not on the structures involved.*

We now list which Preservation Properties hold for which logics.

Theorem 2.
Let $\mathcal{I}$ *be an index structure and* $\mathcal{L}$ *be any of* FOL, $FOL^{m,k}$, $L^\omega_{\omega_1,\omega}$, $L^k_{\omega_1,\omega}$, $MSOL^m$, MTC^m, $MLFP^m$, *or* $FOL[\mathbf{Q}]^{m,k}$ *(*$L_{\omega_1,\omega}[\mathbf{Q}]^k$*) with unary generalized quantifiers. Then* $DJ-PP(\mathcal{L},\mathcal{L})$ *and* $FShu-PP(\mathcal{L},\mathcal{L})$ *hold. Note that this includes* $DJ-PP(FOL^{m,k},FOL^{m,k})$ *and* $FShu-PP(FOL^{m,k},FOL^{m,k})$ *with the same bounds for both arguments, and, similarly, for the other logics.*

Proof: *We first list the cases known from the literature.*

FOL **and** $FOL^{m,k}$**:** *The proofs for* FOL *and* $MSOL$ *are classical; see, in particular [3]. Extension for* $FOL^{m,k}$ *can be done directly from the proof for* FOL.

$MLFP$ **and** $MLFP^m$**:** *The proof for* $MLFP$ *was given in [2].*

$L_{\omega_1,\omega}(\mathbf{Q})^k$**:** *The proof was given in [5].*

Our original proof for MTC^m *is explicitly provided in Appendix D.*

Theorem 3.
Let $\mathcal{L}$ *be any of* FOL, $FOL^{m,k}$, $L^\omega_{\omega_1,\omega}$, $L^k_{\omega_1,\omega}$ $MSOL^m$, MTC^m, $MLFP^m$, *or* $FOL[\mathbf{Q}]^{m,k}$ *with unary generalized quantifiers. There is an algorithm, which produces for given* $\mathcal{L}$, τ_{ind}, $\tau_\alpha,\alpha<\beta$, $\tau_{shuffle}$ *and* $\phi\in\mathcal{L}(\tau_{shuffle})$, *a reduction sequence for* ϕ *for* $(\tau_{ind},\tau_{shuffle})$*-shuffling. However,* F_ϕ *and the* $\psi_{\alpha,j}$ *are exponential in the quantifier rank of* ϕ. *Furthermore,* F *depends on the* $MSOL$*–theory of the index structure restricted to the same quantifier rank as* ϕ.

Proof: *By analyzing the proof of Theorem 2. A special case was analyzed in Gurevich's [18].*
We finally show that our restriction to unary generalized quantifiers (MTC and $MLFP$) is necessary.

Proposition 1. *Theorem 2 does not hold for* 2-TC *or* 2-LFP.

Proof: *Let* $I=\{0,1\}$ *and let the components be finite linear orders. Using a counting argument, it is easy to produce arbitrary large pairs of linear orders* $\mathcal{A}_0$, $\mathcal{A}_1$, *which are* $2\text{-}TC-m$*–equivalent, but of different cardinalities. Now consider the structure* $\mathcal{B}_0=\mathcal{A}_0\sqcup\mathcal{A}_0$ *and* $\mathcal{B}_1=\mathcal{A}_0\sqcup\mathcal{A}_1$. *The* 2-$TC$*–formula, which distinguishes* $\mathcal{B}_0$ *from* $\mathcal{B}_1$ *is the formula* θ, *which asserts that the two components have the same cardinality.* θ *can be written as*

$$2\text{-}TCx_0,x_1,y_0,y_1;first_0,first_1,last_0,last_1(succ_0(x_0,y_0)\wedge succ_1(x_1,y_1)),$$

where $succ_i$ *is the* FOL *formula, expressing the successor in the* i*th component, and* $first_i,last_i$ *are the constant symbols, which are interpreted by the first, respectively, the last element in the* i*th component.*
Now, we discuss various ways of obtaining weighted labeled trees from components, as introduced in [41].

Definition 7 (Finite Disjoint Union of Weighted Labeled Trees)**.**
Let $\tau_i = \langle label^{\tau}_{a\,i}, edge^{\tau}_{i\,i}\rangle$, be a vocabulary of a weighted labeled tree $\mathcal{T}_i$ over Σ. In the general case, the tree over $\Sigma \cup I$ is $\mathcal{T} = \dot{\bigsqcup}_{i\in I}\mathcal{T}_i = \langle \dot{\bigcup}_{i\in I}\mathcal{B}_i, D; label_i(u)(i \in I), label^{\tau}_{a\,i}(i \in I), edge^{\tau}_{i\,\imath}(\imath \in I)\rangle$ for all $i \in I$, where $label_i(u)$ is true iff u came from $\mathcal{B}_i$, I is finite and each element in I is of rank 1.

Now, the following Theorem 4 can be stated, cf. [41].

Theorem 4.
Let I be a finite index set with ℓ elements. Let $\mathcal{T} = \dot{\bigsqcup}_{i\in I}\mathcal{T}_i$ be a weighted labeled tree. Then for every $\varphi \in WMSOL(\tau)$ over boolean semi-rings, there are:

- *a computation over weighted $WMSOL$ formulae*

$$F_{\varphi}(\varpi_{1,1}, \ldots, \varpi_{1,j_1}, \ldots, \varpi_{\ell,1}, \ldots, \varpi_{\ell,j_\ell}), \text{ and}$$

- *$WMSOL$–formulae $\psi_{1,1}, \ldots, \psi_{1,j_1}, \ldots, \psi_{\ell,1}, \ldots, \psi_{\ell,j_\ell}$*

such that for every $\mathcal{T}_i$ and I as above with $\varpi_{i,j} = \varrho_{i,j}$ iff $[\psi_{i,j}] = \varrho_{i,j}$, we have

$$[\varphi] = \varrho \text{ iff } F_{\varphi}(\varpi_{1,1}, \ldots, \varpi_{1,j_1}, \ldots, \varpi_{\ell,1}, \ldots, \varpi_{\ell,j_\ell}) = \varrho.$$

Moreover, F_{φ} and the $\psi_{i,j}$ are computable from φ, ℓ and vocabularies alone, but are exponential in the quantifier rank of φ.

In Theorem 5, we also list some other options of commutative semi-rings to choose.

Theorem 5. *In addition, the following semi-rings satisfy Theorem 4:*

- **Subset Semi-ring:** $(P(A), \cap, \cup, \emptyset, A)$**.** *The proof by analyzing and extension of the proof in [14].*
- **Fuzzy Semi-ring:** $([0,1], \vee, \wedge, 0, 1)$*. The proof by analyzing and extension of the proof in [31].*
- **Extended natural number:** $(\mathbf{N} \cup \{\infty\}, +, \cdot, 0, 1)$**.** *The proof by analyzing and extension of the proof in [31].*
- **Tropical Semi-ring:** $(\mathbf{R}_+ \cup \{+\infty\}, \min, +, +\infty, 0)$**,** *cf. [41].*
- **Arctic Semi-ring:** $(\mathbf{R}_+ \cup \{-\infty\}, \max, +, -\infty, 0)$**.** *The proof by analyzing and extension of the proof in [41].*

4 Syntactically Defined Translation Schemes

The second logical reduction technique that we use is *the syntactically defined translation schemes*, which describe transformations of logical structures. The notion of abstract translation schemes comes back to Rabin, cf. [38]. They give rise to two induced maps: translations and transductions. *Transductions* describe the induced transformation of logical structures and the *translations* describe the induced transformations of logical formulae.

Definition 8 (Translation Schemes Φ)**.**
Let τ_1 and τ_2 be two vocabularies and $\mathcal{L}$ be a logic. Let $\tau_2 = \{R_1, \ldots, R_m\}$ and let $\rho(R_i)$ be the arity of R_i. Let $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m\rangle$ be formulae of $\mathcal{L}(\tau_1)$. Φ is κ–feasible for τ_2 over τ_1 *if φ has exactly κ distinct free variables and each ψ_i has $\kappa\rho(R_i)$ distinct free variables. Such a $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m\rangle$ is also called a* κ–τ_1–τ_2–translation scheme *or, shortly, a* translation scheme*, if the parameters are clear in the context.*

The above definition as a rule assumes *one sorted* logical structures. However, if we deal with weighted logics like $WMSOL$ this is not a case. In general, if $\mathcal{L}$ is defined over particular kinds of $\mathcal{L}$-objects (like graphs, words, etc.), then the exact logical presentation of the objects must be explicitly provided. $WMSOL$ is defined over the weighted labeled trees and we present them as logical structures in the following way:

Definition 9 (Weighted Labeled Tree over a ptv-monoid D)**.**
Given a ptv-monoid D, the weighted labeled tree over D is the following logical many-sorted structure $\mathcal{T} = \langle \mathcal{B}, D; label_a, edge_i \rangle$, where

Universe of $\mathcal{T}$ *is many-sorted: $\mathcal{B}$ is the tree domain and D comes from the monoid.*

Relations of $\mathcal{T}$ *are defined as following. $label_a$ is a unary relation, which for each $a \in \Sigma$ means that $u \in \mathcal{B}$ is labeled by a, and $edge_i$ is a binary relation, which for each $1 \le i \le \max_\Sigma$ and two $u_1, u_2 \in \mathcal{B}$ means that u_2 is an immediate prefix of u_1.*

In the context of $WMSOL$, Definition 8 may be paraphrased as follows:

Definition 10 (Translation Schemes Φ_D on Weighted Labeled Trees)**.**
Let τ_1 and τ_2 be two vocabularies of weighted labeled trees. Let $\tau_1 = \langle label_a^{\tau_1}, edge_i^{\tau_1} \rangle$, over ptv-monoid D. Let $\Phi = \langle \phi_{\mathcal{B}}, \phi_D; \psi_{label_a}, \psi_{edge_i} \rangle$ be an almost boolean WMSOL formulae (for each $a \in \Sigma$ and $1 \le i \le \max_\Sigma$). We say that Φ_D is feasible for τ_2 over τ_1 *if*

- *$\phi_{\mathcal{B}}$ has exactly 1 distinct free first order variable over $\mathcal{B}$,*
- *ϕ_D is a tautology with exactly one free variable over D,*
- *each ψ_{label_a} has exactly 1 distinct free first order variable over $\mathcal{B}$,*
- *each ψ_{edge_i} has exactly 2 distinct free first order variables over $\mathcal{B}$.*

In general, Definition 8 must be adopted to the given logic $\mathcal{L}$, if it is not straightforward.

For $\mathcal{L}$ like FOL, $MSOL$, MTC, $MLFP$ or FOL with unary generalized quantifiers, with a translation scheme Φ we can *naturally* associate a (partial) function Φ^* from τ_1–structures to τ_2–structures.

Definition 11 (Induced map Φ^*)**.**
Let $\mathcal{A}$ be a τ_1–structure with universe A and Φ be κ–feasible for τ_2 over τ_1. The structure $\mathcal{A}_\Phi$ is defined as follows:

1. *The universe of $\mathcal{A}_\Phi$ is the set $A_\Phi = \{\bar{a} \in A^\kappa : \mathcal{A} \models \varphi(\bar{a})\}$.*
2. *The interpretation of R_i in $\mathcal{A}_\Phi$ is the set*

$$A_\Phi(R_i) = \{\bar{a} \in {A_\Phi}^{\rho(R_i)\cdot\kappa} : \mathcal{A} \models \psi_i(\bar{a})\}.$$

   Note that $\mathcal{A}_\Phi$ is a τ_2–structure of cardinality at most $\mid A \mid^\kappa$.
3. *The partial function $\Phi^* : Str(\tau_1) \to Str(\tau_2)$ is defined by $\Phi^*(\mathcal{A}) = \mathcal{A}_\Phi$. Note that $\Phi^*(\mathcal{A})$ is defined iff $\mathcal{A} \models \exists \bar{x} \varphi$.*

The case of $WMSOL$ it is a bit more tricky. The (partial) function Φ_D^* from τ_1–trees to τ_2–trees is defined as follows:

Definition 12 (Induced map Φ_D^*)**.**
Let $\mathcal{T}^{\tau_1}$ be a τ_1–tree and Φ_D be feasible for τ_2 over τ_1. The structure $\mathcal{T}_{\Phi_D}^{\tau_2}$ is defined as follows:

• **The many-sorted universe** $\mathcal{B}, D$**:** *are the sets:*

1. $\mathcal{B}_{\Phi_D} = \{u \in \mathcal{B}^{\tau_1} : \mathcal{T}^{\tau_1} \models \phi_{\mathcal{B}}(u)\}$;
2. $D_{\Phi_D} = D$.

- **Relations** $label_a^{\tau_2}, edge_i^{\tau_2}$**:** *For each* $a \in \Sigma$ *and* $1 \le i \le \max_\Sigma$*:*
  1. *The interpretation of each* $label_a$ *in* $\mathcal{T}^{\tau_2}{}_{\Phi_D}$ *is the set*
  $$\mathcal{T}^{\tau_2}{}_{\Phi_D}(label_a) = \{u \in \mathcal{B}^{\tau_1} : \mathcal{T}^{\tau_1} \models \psi_{label_a}(u)\};$$
  2. *The interpretation of each* $edge_i$ *in* $\mathcal{T}^{\tau_2}{}_{\Phi_D}$ *is the set of pairs*
  $$\mathcal{T}^{\tau_2}{}_{\Phi_D}(edge_i) = \{(u_1, u_2) \in \mathcal{B}^{\tau_1 2} : \mathcal{T}^{\tau_1} \models \psi_{edge_i}(u_1, u_2)\};$$
- $\Phi_D{}^*$**:** *The partial function* $\Phi_D{}^* : Trees(\tau_1) \to Trees(\tau_2)$ *is defined by*
$$\Phi_D{}^*(\mathcal{T}^{\tau_1}) = \mathcal{T}^{\tau_2}{}_{\Phi_D}.$$

Note that $\Phi_D{}^*(\mathcal{T}^{\tau_1})$ *is defined iff* $\mathcal{T}^{\tau_1} \models \phi_{\mathcal{B}}(u)$.

Again, for $\mathcal{L}$ like FOL, $MSOL$, MTC, $MLFP$ or FOL with unary generalized quantifiers, with a translation scheme Φ we can also *naturally* associate a function $\Phi^\#$ from $\mathcal{L}(\tau_2)$–formulae to $\mathcal{L}(\tau_1)$–formulae.

Definition 13 (Induced map $\Phi^\#$)**.**
Let θ *be a* τ_2*–formula and* Φ *be* κ*–feasible for* τ_2 *over* τ_1*. The formula* θ_Φ *is defined inductively as follows:*

1. *For* $R_i \in \tau_2$ *and* $\theta = R(x_1, \ldots, x_m)$ *let* $x_{j,h}$ *be new variables with* $i \le m$ *and* $h \le \kappa$ *and denote by* $\bar{x}_i = \langle x_{i,1}, \ldots, x_{i,\kappa}\rangle$*. We put* $\theta_\Phi = \psi_i(\bar{x}_1, \ldots, \bar{x}_m)$.
2. *For the boolean connectives the translation distributes, i.e., if* $\theta = (\theta_1 \vee \theta_2)$ *then* $\theta_\Phi = (\theta_{1\Phi} \vee \theta_{1\Phi})$ *and if* $\theta = \neg\theta_1$ *then* $\theta_\Phi = \neg\theta_{1\Phi}$*, and similarly for* $\wedge$.
3. *For the existential quantifier, we use relativization, i.e., if* $\theta = \exists y\theta_1$*, let* $\bar{y} = \langle y_1, \ldots, y_\kappa\rangle$ *be new variables. We put* $\theta_\Phi = \exists\bar{y}(\varphi(\bar{y}) \wedge \theta_{1\Phi})$.
4. *For (monadic) second order variables* U *of arity* ℓ *(*$\ell = 1$ *for* $MSOL$*) and* $\bar{v}$ *a vector of length* ℓ *of first order variables or constants we translate* $U(\bar{v})$ *by treating* U *like a relation symbol above and put*
$$\theta_\Phi = \exists V(\forall\bar{v}(V(\bar{v}) \to (\phi(\bar{v_1}) \wedge \ldots \phi(\bar{v_\ell}) \wedge (\theta_1)_\Phi))).$$
5. *For generalized quantifiers, if* $\theta = Q_i v^1, v^2, \ldots, v^m\theta_1(v^1, v^2, \ldots, v^m, \ldots)$*, then let* $\bar{v}^j = \langle v_1^j, \ldots, v_k^j\rangle$ *be new variables for* v^j*. We set*
$$\theta_\Phi = Q_i\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m(\theta_1(\bar{v}^1, \bar{v}^2, \ldots, \bar{v}^m, \ldots)_\Phi).$$
6. *For infinitary logics, if* $\theta = \bigwedge\Psi$ *then* $\theta_\Phi = \bigwedge\Psi_\Phi$.
7. *For* LFP*, if* $\theta = n\text{-}LFP\bar{x}, \bar{y}, \bar{u}, \bar{v}\theta_1$ *then* $\theta_\Phi = (n \cdot \kappa)\text{-}LFP\bar{x}, \bar{y}, \bar{u}, \bar{v}\theta_{1\Phi}$.
8. *For* TC*, if* $\theta = n\text{-}TC\bar{x}, \bar{y}, \bar{u}, \bar{v}\theta_1$ *then* $\theta_\Phi = (n \cdot \kappa)\text{-}TC\bar{x}, \bar{y}, \bar{u}, \bar{v}\theta_{1\Phi}$.
9. *For weighted formulae over ptv-monoid* D*:*
   (a) *for* d *we do nothing;*
   (b) *for a boolean formula* β *we put* $\zeta_{\Phi_D} = \beta$;

(c) for boolean connectives and quantifiers the translation distributes.

10. *The function* $\Phi^{\#} : \mathcal{L}(\tau_2) \to \mathcal{L}(\tau_1)$ *is defined by* $\Phi^{\#}(\theta) = \theta_{\Phi}$.

Note that the case of weighted formulae over ptv-monoid D is the most complicated in the definition. In general, given $\mathcal{L}$, Definition 13 must be adopted to all well-formed formulae of this logic.

The following fundamental theorem is easily verified for correctly defined $\mathcal{L}$ translation schemes. Its origins go back at least to the early years of modern logic, cf. [19, page 277 ff].

Theorem 6. *Let* $\Phi = \langle \varphi, \psi_1, \ldots, \psi_m \rangle$ *be a* κ*–*τ_1*–*τ_2*–translation scheme,* $\mathcal{A}$ *a* τ_1*-structure and* θ *a* $\mathcal{L}(\tau_2)$*–formula. Then*

$$\mathcal{A} \models \Phi^{\#}(\theta) \text{ iff } \Phi^{*}(\mathcal{A}) \models \theta.$$

5 Strongly Distributed Structures

The disjoint union and shuffles as such are not very interesting. However, combining it with translation schemes gives as a rich repertoire of composition techniques. Now, we generalize the disjoint union or shuffling of structures to *strongly distributed structures* in the following way:

Definition 14 (Strongly Distributed Structures)**.**
Let $\mathcal{I}$ *be a finitely partitioned index structure and* $\mathcal{L}$ *be any like FOL, MSOL, WMSOL, MTC, MLFP, or FOL with unary generalized quantifiers. Let* $\mathcal{A} = \bigsqcup_{i \in I} \mathcal{A}_i$ *be a* τ*–structure, where each* $\mathcal{A}_i$ *is isomorphic to some* $\mathcal{B}_1, \ldots, \mathcal{B}_\beta$ *over the vocabularies* $\tau_1, \ldots, \tau_\beta$*, in accordance with the partition. For a scalar (non–vectorized)* τ_1*–*τ_2 $\mathcal{L}$*–translation scheme* Φ*, the* Φ–Strongly Distributed Structure*, which is composed from* $\mathcal{B}_1, \ldots, \mathcal{B}_\beta$ *over* $\mathcal{I}$*, is the structure* $\Phi^{*}(\mathcal{A})$*, or rather any structure isomorphic to it.*

Now, our main theorem can be formulated as follows:

Theorem 7.
Let $\mathcal{I}$ *be a finitely partitioned index structure,* $\mathcal{L}$ *be any of FOL, MSOL, MTC, MLFP, MSOL or FOL with unary generalized quantifiers. Let* $\mathcal{S}$ *be a* Φ*–Strongly Distributed Structure, composed from* $\mathcal{B}_1, \ldots, \mathcal{B}_\beta$ *over* $\mathcal{I}$*, as above. For every* $\phi \in \mathcal{L}(\tau)$ *there are*

1. *a boolean function* $F_{\Phi,\phi}(b_{1,1}, \ldots, b_{1,j_1}, \ldots, b_{\beta,1}, \ldots, b_{\beta,j_\beta}, b_{I,1}, \ldots, b_{I,j_I})$,
2. $\mathcal{L}$*–formulae* $\psi_{1,1}, \ldots, \psi_{1,j_1}, \ldots, \psi_{\beta,1}, \ldots, \psi_{\beta,j_\beta}$ *and*
3. *MSOL–formulae* $\psi_{I,1}, \ldots, \psi_{I,j_I}$

such that for every $\mathcal{S}$*,* $\mathcal{I}$ *and* $\mathcal{B}_i$ *as above with* $\mathcal{B}_i \models \psi_{i,j}$ *iff* $b_{i,j} = 1$ *and* $\mathcal{I} \models \psi_{I,j}$ *iff* $b_{I,j} = 1$ *we have*

$$\mathcal{S} \models \phi \quad \text{iff} \quad F_{\Phi,\phi}(b_{1,1}, \ldots, b_{1,j_1}, \ldots, b_{\beta,1}, \ldots, b_{\beta,j_\beta}, b_{I,1}, \ldots, b_{I,j_I}) = 1.$$

$F_{\Phi,\phi}$ *and* $\psi_{i,j}$ *are computable from* $\Phi^{\#}$ *and* ϕ*, but they are exponential in the quantifier rank of* ϕ*.*

Proof: *By analyzing the proof of Theorem 3 and using Theorem 6.*
Moreover, in [41], the following was proven for $WMSOL$:

Theorem 8.
Let $\mathcal{I}$ be a finite index structure and let $\mathcal{T}$ be Φ-Strongly Distributed over $\mathcal{T}_1$,...,$\mathcal{T}_\ell$ over I, as above. For every $\varphi \in WMSOL(\tau)$ that satisfies Theorem 4 there are:

- *a computation over weighted formulae $F_{\Phi,\varphi}(\varpi_{1,1},\dots,\varpi_{1,j_1},\dots,\varpi_{\ell,1},\dots,\varpi_{\ell,j_\ell})$, and*
- *$WMSOL$–formulae $\psi_{1,1},\dots,\psi_{1,j_1},\dots,\psi_{\ell,1},\dots,\psi_{\ell,j_\ell}$*

such that for every $\mathcal{T}_i$ and $\mathcal{I}$ as above with $\varpi_{i,j} = \varrho_{i,j}$ iff $[\psi_{i,j}] = \varrho_{i,j}$ we have

$$[\varphi] = \varrho \text{ iff } F_{\Phi,\varphi}(\varpi_{1,1},\dots,\varpi_{1,j_1},\dots,\varpi_{\ell,1},\dots,\varpi_{\ell,j_\ell}) = \varrho.$$

Moreover, $F_{\Phi,\varphi}$ and $\psi_{i,j}$ are computable from $\Phi^{\#}$ and φ, but they are exponential in the quantifier rank of φ.

6 Complexity Analysis

In this section, we discuss under what conditions our approach improves the complexity of computations, when measured in the size of the composed structures only. A strongly distributed structure (weighted tree) is now submitted to a computation unit and we want to know: how long does it take to check whether ϕ is true on the structure. Now, we give the general complexity analysis of the computation on strongly distributed structures.

Assume that $\mathcal{A}$ is a strongly distributed structure. Its components are $\mathcal{A}_\imath$ with index structure $\mathcal{I}$ ($i \in I$), and we want to check whether ϕ is true in $\mathcal{A}$. Assume that:

- $\mathcal{T}(N)$ or $\mathcal{T}_{old}(N)$ denotes the time to solve the problem by the traditional sequential way (here, N denotes the size of the coding of $\mathcal{A}$);
- $\mathcal{E}_\mathcal{I}$ denotes the time to extract index structure $\mathcal{I}$ from $\mathcal{A}$;
- $\mathcal{E}_\imath$ denotes the time to extract each $\mathcal{A}_\imath$ from G;
- $\mathcal{C}_\mathcal{I}(n_I)$ denotes the time to compute all values of $b_{I,j}$, where n_I is the size of I;
- $\mathcal{C}_\imath(n_\imath)$ denotes the time to compute all values of $b_{\imath,j}$, where $n_\imath$ is the size of $A_\imath$;
- $\mathcal{T}_{F_{\Phi,\phi}}$ denotes the time to build $F_{\Phi,\phi}$;
- $\mathcal{T}_\mathcal{S}$ denotes the time to achieve one result of $F_{\Phi,\phi}$.

According to these symbols, the new computation time is:

$$\mathcal{T}_{new} = \mathcal{E}_\mathcal{I} + \Sigma_{\imath\in I}\mathcal{E}_\imath + \mathcal{C}_\mathcal{I} + \Sigma_{\imath\in I}\mathcal{C}_\imath + \mathcal{T}_{F_{\Phi,\phi}} + \mathcal{T}_\mathcal{S}$$

and the question to answer is: When does hold $\mathcal{T}_{old} > \mathcal{T}_{new}$? For more details, cf. [39].

7 Conclusion and Outlook

In this contribution, we introduced the notion of strongly distributed structures and presented a uniform approach to incremental automated reasoning on such structures. The approach is based on a systematic use of two logical reduction techniques: *Feferman-Vaught reductions* and *the syntactically defined translation schemes*.

Our general scenario is as follows: given logic $\mathcal{L}$, structure $\mathcal{A}$ as a composition of structures $\mathcal{A}_i, i \in I$, index structure $\mathcal{I}$ and formula ϕ of the logic to be evaluated on $\mathcal{A}$. The question is: What is the reduction sequence of ϕ, if any? We propose a general approach to try to answer the question and to investigate the computation gain of the incremental evaluations. The general template is defined as follows:

1. **Prove preservation theorems**
   Given logic $\mathcal{L}$.
   (a) **Define disjoint union of $\mathcal{L}$-structures** The logic may be defined for arbitrary structures or rather for a class of structures like graphs, (directed) acyclic graphs, trees, words, (Mazurkiewicz) traces, cf. [6], or (lossy) message sequence charts, etc. In the general case, we use Definition 1 that provides a logical definition of *disjoint union* of the components: $\mathcal{A} = \dot{\bigcup}_{i\in I}\mathcal{A}_i$. An adaptation of Definition 1 to the case of Weighted Monadic Second Order Logic ($WMSOL$), which is introduced over trees, is presented in Definition 7. If logic $\mathcal{L}$ is introduced over another class of structures, then Definition 1 must be aligned accordingly.
   (b) **Define a preservation property $XX-PP$ for $\mathcal{L}$** After we defined the appropriate disjoint union of structures, we define the notion of a *(XX) preservation property (PP)* for logics; see Definitions 3 and 4.
   (c) **Prove the preservation property $XX-PP$ for $\mathcal{L}$** Now, we try to prove the corresponding preservation property for $\mathcal{L}$. As a rule, such preservation theorem can be proven by suitable Pebble games, which are generalizations of Ehrenfeucht-Fraïssé games. This gives usually a version of $XX-PP(\mathcal{L}_1^{m_1,k_1},\mathcal{L}_2^{m_2,k_2})$ for suitable chosen definitions of quantifier rank (m_j) and counting of variables (k_j). Our Theorem 2 shows that the preservation theorems hold for FOL, $FOL^{m,k}$, $L^{\omega}_{\omega_1,\omega}$, $L^{k}_{\omega_1,\omega}$, $MSOL^m$, MTC^m, $MLFP^m$, or $FOL[\mathbf{Q}]^{m,k}$ ($L_{\omega_1,\omega}[\mathbf{Q}]^k$) with unary generalized quantifiers. However, theorems like Theorem 2 are not always true. In Proposition 1, we show that our restriction to unary generalized quantifiers (MTC and $MLFP$) is necessary. In fact, Theorem 2 do not hold for 2-TC or 2-LFP. Moreover, Theorem 5 shows for which semi-rings $WMSOL$ guaranties the preservation property.
2. **Define Translation Schemes**
   Given logic $\mathcal{L}$.
   Definitions 8 introduces the classical *syntactically defined translation schemes*. Definition 9 is an adaption of Definitions 8 to the case of a many-sorted structures. In general, Definitions 8 must be adopted in the similar way to the given logic $\mathcal{L}$. Definitions 8 gives rise to two induced maps, translations and transductions. Transductions describe the induced transformation of $\mathcal{L}$-structures and the translations describe the induced transformations of $\mathcal{L}$-formulae: see Definitions 11, 12 and 13. Again, the presented adaptation of the definitions to the case of $WMSOL$ is a bit tricky. If the $\mathcal{L}$-translation scheme is defined correctly then the proof of the corresponding variation of Theorem 6 is easily verified.
3. **Strongly Distributed Structures**
   Given $\mathcal{L}$-structure $\mathcal{A}$.
   In this step, we defined disjoint unions (and shuffles) of $\mathcal{L}$-structures. However, as such they are not very interesting. On the other hand, combining them with translation schemes gives as a rich repertoire of composition techniques. Using translation scheme Φ, we introduce the notion of *strongly distributed structures*. If the given $\mathcal{L}$-structure $\mathcal{A}$ is a Φ-strongly distributed composition of its components then we may apply $\mathcal{L}$-variation of our main Theorem 7 to it. Theorem 7 shows how to effectively compute the reduction sequences for different logics, under investigation, for the strongly distributed structures.

Finally, we derive a method for evaluating $\mathcal{L}$-formula ϕ on $\mathcal{A}$, which is a Φ-strongly distributed composition of its components. The method proceeds as follows:

Preprocessing: Given ϕ and Φ, but not a $\mathcal{A}$, we construct a sequence of formulas $\psi_{i,j}$ and an evaluation function $F_{\Phi,\phi}$ as in Theorem 7.

Incremental Computation: We compute the local values $b_{i,j}$ for each component of the $\mathcal{A}$.

Solution: Theorem 7 now states that ϕ, expressible in the corresponding logic $\mathcal{L}$, on $\mathcal{A}$ may be effectively computed from $b_{i,j}$, using $F_{\Phi,\phi}$.

We showed that the approach works for lots of extensions of FOL but not all. The considered extensions of FOL are suitable candidates for modeling languages for components and services, used in incremental automated reasoning, data mining, decision making, planning and scheduling.

We plan to apply the proposed methodology to the incremental reasoning, based on the promising variations of $WMSOL$ as introduced recently in [25], [26], [36].

References

[1] B. Bollig and P. Gastin. Weighted versus probabilistic logics. In V. Diekert and D. Nowotka, editors, *DLT 2009, LNCS R5583*, pages 18–38, Berlin Heidelberg, 2009. Springer-Verlag.

[2] U. Bosse. *Ehrenfeucht–Fraïssé Games for Fixed Point Logic*. PhD thesis, Department of Mathematics, University of Freiburg, Germany, 1995.

[3] C.C. Chang and H.J. Keisler. *Model Theory*. Studies in Logic, vol 73. North–Holland, 3rd edition, 1990.

[4] A. Cyriac and P. Gastin. Reasoning about distributed systems: WYSIWYG (invited talk). In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, pages 11–30, 2014.

[5] A. Dawar and L. Hellat. The expressive power of finitely many genaralized quantifiers. Technical Report CSR 24–93, Computer Science Department, University of Wales, University College of Swansea,U.K, 1993.

[6] V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific, 1995.

[7] M. D'Inverno, M. Luck, M. Georgeff, D. Kinny, and M. Wooldridge. The dMARS architecture: A specification of the distributed multi-agent reasoning system. *Autonomous Agents and Multi-Agent Systems*, 9(1-2):5–53, 2004.

[8] M. Droste and P. Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380:69–86, 2007.

[9] M. Droste, D. Götze, S. Märcker, and I. Meinecke. Weighted tree automata over valuation monoids and their characterization by weighted logics. In W. Kuich and G. Rahonis, editors, *Bozapalidis Festschrift, LNCS 7020*, pages 30–55, Berlin Heidelberg, 2011. Springer-Verlag.

[10] M. Droste and I. Meinecke. Describing average- and longtime-behavior by weighted MSO logics. In P. Hliněný and A. Kučera, editors, *MFCS 2010, LNCS 6281*, pages 537–548, Berlin Heidelberg, 2010. Springer-Verlag.

[11] M. Droste and H. Vogler. Weighted logics for unranked tree automata. *Theory of Computing Systems*, 48(1):23–47, 2009.

[12] H.D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic, 2nd edition*. Undergraduate Texts in Mathematics. Springer-Verlag, 1994.

[13] R. Fagin. Generalized first-order spectra and polynomial time recognizable sets. In R. Karp, editor, *Complexity of Computation*, volume 7 of *American Mathematical Society Proc*, pages 27–41. Society for Industrial and Applied Mathematics, 1974.

[14] S. Feferman and R. Vaught. The first order properties of products of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

[15] I. Fichtner. Weighted picture automata and weighted logics. In *STACS 2006*, pages 313–324. Springer, 2006.

[16] M.G. Garey and D.S. Johnson. *Computers and Intractability.* Mathematical Series. W.H. Freeman and Company, 1979.

[17] E. Grädel. On transitive closure logic. In E. Börger, G. Jäger, H. Kleine Büning, and M.M. Richter, editors, *Computer Science Logic*, volume 626 of *Lecture Notes in Computer Science*, pages 149–163. Springer Verlag, 1992.

[18] Y. Gurevich. Modest theory of short chains, I. *Journal of Symbolic Logic*, 44:481–490, 1979.

[19] D. Hilbert and P. Bernays. *Grundlagen der Mathematik, I*, volume 40 of *Die Grundleheren der mathematischen Wissenschaften in Einzeldarstellungn.* Springer Verlag, Heidelberg, 2nd edition, 1970.

[20] IBM. Big data and information management. http://www-01.ibm.com/software/data/bigdata/, 2014.

[21] N. Immerman. Relational queries computable in polynomial time. In *STOC'82*, pages 147–152. ACM, 1982.

[22] N. Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, Aug. 1987.

[23] A. Jacobs. The pathologies of Big Data. *Commun. ACM*, 52(8):36–44, August 2009.

[24] P. G. Kolaitis and J. A. Väänänen. Generalized quantifiers and pebble games on finite structures. In *LiCS'92*, pages 348–359. IEEE, 1992.

[25] S. Kreutzer and C. Riveros. Quantitative monadic second-order logic. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 113–122, 2013.

[26] N. Labai and J.A. Makowsky. Weighted automata and monadic second order logic. In *Proceedings Fourth International Symposium on Games, Automata, Logics and Formal Verification, GandALF 2013, Borca di Cadore, Dolomites, Italy, 29-31th August 2013.*, pages 122–135, 2013.

[27] P. Lindström. First order predicate logic with generalized quantifiers. *Theoria*, 32:186–195, 1966.

[28] P. Lindström. On extensions of elementary logic. *Theoria*, 35:1–11, 1969.

[29] J.A. Makowsky and Y.B. Pnueli. Oracles and quantifiers. In *Computer Science Logic, 7th Workshop, CSL '93, Swansea, United Kingdom, September 13-17, 1993, Selected Papers*, pages 189–222, 1993.

[30] J.A. Makowsky and Y.B. Pnueli. Arity vs. alternation in second order definability. In *LFCS'94*, volume 813 of *Lecture Notes in Computer Science*, pages 240–252. Springer, 1994. (Extended version to appear in the Annals of Pure and Applied Logic, 1995).

[31] J.A. Makowsky and E.V. Ravve. Incremental model checking for decomposable structures. In *Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 540–551. Springer Verlag, 1995.

[32] E. Mandrali and G. Rahonis. Recognizable tree series with discounting. *Acta Cybernetica*, 19(2):411–439, 2009.

[33] C. Mathissen. Definable transductions and weighted logics for texts. In *Proceedings of the 11th International Conference on Developments in Language Theory*, DLT'07, pages 324–336, Berlin, Heidelberg, 2007. Springer-Verlag.

[34] C. Mathissen. Weighted logics for nested words and algebraic formal power series. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 221–232, Berlin, Heidelberg, 2008. Springer-Verlag.

[35] I. Meinecke. Weighted logics for traces. In *Proceedings of the First International Computer Science Conference on Theory and Applications*, CSR'06, pages 235–246, Berlin, Heidelberg, 2006. Springer-Verlag.

[36] B. Monmege. *Spécification et Vérification de Propriétés Quantitatives: Expressions, Logiques et Automates.* PhD thesis, Laboratoire Spécification et Vérification, École Normale Supérieure de Cachan, Cedex, France, 2013.

[37] A. Mostowsky. On a generalization of quantifiers. *Fundamenta Mathematicae*, 44:12–36, 1957.

[38] M.O. Rabin. A simple method for undecidability proofs and some applications. In Y. Bar Hillel, editor, *Logic, Methodology and Philosophy of Science II*, Studies in Logic, pages 58–68. North Holland, 1965.

[39] E.R. Ravve and Z. Volkovich. Four scenarios of effective computations on sum-like graphs. In *Proc. of the The 9th Intern. Multi-Conference on Computing in the Global Informationin Technology*, pages 1–8, 2014.

[40] E.V. Ravve. Views and updates over distributed databases. In F. Winkler, V. Negru, T. Ida, T. Jebelean, D. Petcu, S.M. Watt, and D. Zaharie, editors, *16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014, Timisoara, Romania, September 22-25, 2014*, pages 341–348. IEEE, 2014.

[41] E.V. Ravve, Z. Volkovich, and G.-W. Weber. Effective optimization with weighted automata on decomposable trees. *Optimization Journal, Special Issue on Recent Advances in Continuous Optimization on the Occasion of the 25th European Conference on Operational Research (EURO XXV 2012)*, 63:109–127, 2014.

[42] A. Tarski. A model-theoretical result concerning infinitary logics. *Notices of the American Mathematical Society*, 8:260–280, 1961.

[43] M. Vardi. The complexity of relational query languages. In *STOC'82*, pages 137–146. ACM, 1982.

A Motivating Example: Airlines of Two Countries

The example is taken verbatim from [40]. We are given two countries A_1 and A_2 with their local airline connections. These are represented by two disjoint undirected graphs, with the AIRPORTS as vertices V_{A_1} and V_{A_2}, and (possibly labeled) edges for the connections (where the labels indicate AIRLINE, DAY, etc.). The sets of edges are denoted by E_{A_1} and E_{A_2} (using several edge relations in the labeled case such as $E_{A_i}^{AIRLINE}$ and $E_{A_i}^{DAY}$, etc). Each of these graphs is stored in a different place. If we add now different unary predicates in each country to mark the International Airports (P_{A_1} and P_{A_2}) and we stipulate that all A_1 International Airports be connected with all A_2 International Airports, then we get the situation, abstractly described below, see Figure 3.

Now, we discuss how the properties *connectivity* and *existence of cycles* can be reduced to properties of the two labeled graphs and the the definition of the connections between them (via a translation scheme). We note that the properties connectivity and existence of cycles are **not** First Order Logic (FOL) definable.

Assume we are given two undirected finite graphs $\mathcal{A}_1 = \langle V_{A_1}, E_{A_1}, P_{A_1}\rangle$ and $\mathcal{A}_2 = \langle V_{A_2}, E_{A_2}, P_{A_2}\rangle$, where V_{A_i} denotes a set of vertices, E_{A_i} denotes a set of edges and P_{A_1}, P_{A_2} are one place relations (labels, vertex colourings), respectively. Let $\mathcal{A}$ be the disjoint union of $\mathcal{A}_1$ and $\mathcal{A}_2$ with additional edges forming a complete bipartite graph on the coloured vertices. We define this composition of two coloured graphs formally as follows: $\mathcal{A} = \mathcal{A}_1 \bigoplus \mathcal{A}_2 = \langle A_1 \dot\cup A_2, E\rangle$, where $A_1 \dot\cup A_2$ denotes disjoint union of sets of vertices, and two vertices x and y of A belongs to E iff $\Phi_E(x,y)$ holds, where

$$\Phi_E(x,y) = ((x,y) \in E_{A_1} \vee (x,y) \in E_{A_2}) \bigvee ((x \in P_{A_1} \wedge y \in P_{A_2}) \vee (x \in P_{A_2} \wedge y \in P_{A_1})).$$

The result is a graph with two kinds of labels. Clearly $\Phi_E(x,y)$ can be rewritten as a FOL formula in the vocabulary of labeled graphs. Assume that we want to check whether $\mathcal{A}$ is connected or has cycles.

Connectivity: The following property is easily seen:

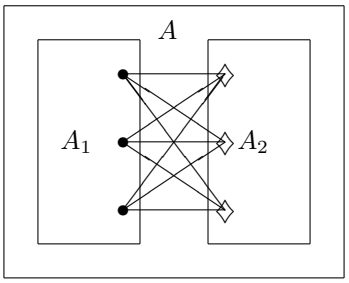

Figure 3: Composition of two graphs: A_1 and A_2.

(*) $\mathcal{A}$ is connected iff both in $\mathcal{A}_1$ and $\mathcal{A}_2$ it is true that every connected component has at least one coloured vertex.

We observe the following:

- Connectivity can be expressed by a formula of Monadic Second Order Logic ($MSOL$) φ_{conn}.
- The property that every connected component has at least one coloured vertex can be expressed by formula of $MSOL$ ψ.
- Φ_E can be expressed by a formula of $MSOL$ over the disjoint union of $\mathcal{A}_1$ and $\mathcal{A}_2$. Actually, in this example, $\Phi_E(x, y)$ is a quantifier free FOL formula.
- To check whether $\mathcal{A}$ is connected ($\mathcal{A} \models \varphi_{conn}$) it suffices to check, using (*), that $\mathcal{A}_1 \models \psi$ and $\mathcal{A}_2 \models \psi$.
- Moreover, by defining boolean values $b_i = 1$ iff $\mathcal{A}_i \models \psi$ there is a boolean function F such that $\mathcal{A} \models \varphi_{conn}$ iff $F(b_1, b_2) = 1$.
- To check whether $\mathcal{A}$ is not connected ($\mathcal{A} \models \neg\varphi_{conn}$) it suffices to evaluate F again and to check that $F(b_1, b_2) = 0$.
- The formula ψ and the boolean function F depend only on the syntactic structure of Φ_E and φ_{conn}, but not on the structures $\mathcal{A}_i$.
- If we have checked the connectivity of $\mathcal{A}$ and now wish to check the connectivity of $\mathcal{A}' = \mathcal{A}'_1 \bigoplus \mathcal{A}_2$ for a different A'_1 we just have to recompute $\mathcal{A}'_1 \models \psi$, and $F(b'_1, b_2)$ for $b'_1 = 1$ iff $\mathcal{A}'_1 \models \psi$, but we do not have to recompute b_2.

Cyclicity: To check whether $\mathcal{A}$ has cycles we observe that

(†) $\mathcal{A}$ has a cycle iff $\mathcal{A}_1$ has a cycle, or $\mathcal{A}_2$ has a cycle, or there are at least two connected coloured vertices in $\mathcal{A}_{2-i}$ and at least one coloured vertex in $\mathcal{A}_{i+1}$, where $i \in \{0, 1\}$,

and proceed similarly as follows.

- We first write the property as a formula φ_{cycle} in $MSOL$.
- Then, using (†), which depends only on φ_{cycle} and Φ_E, we look for formulas $\psi_{1,1}, \ldots, \psi_{1,n_1}$ and $\psi_{2,1}, \ldots, \psi_{2,n_2}$ in $MSOL$, which will give us the properties to be checked in $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively.

- Then, again using (†), we look for a boolean function F in n_1+n_2 arguments $b_{1,1},\ldots,b_{2,n_2}$.
- Now, we put $b_{i,j} = 1$ iff $\mathcal{A}_i \models \psi_{i,j}$ and hope to conclude that $\mathcal{A} \models \varphi_{cycle}$ iff $F(b_{1,1},\ldots,b_{2,n_2}) = 1$.

Observations

1. In the above example, the construction has three inputs: the two coloured graphs and the formula Φ_E, which defines the new edge relation on the disjoint union. The formula Φ_E can be viewed as a parameter in the definition of a binary operation $\bigoplus_\Phi(\mathcal{A}_1,\mathcal{A}_2)$ on coloured graphs. The general case is obtained, using Translation Schemes, as described in Section 4.
2. In the example above, we have the index set $I = \{1,2\}$, and the properties φ_{conn}, which says that the graph is connected, and ψ, which says that every connected component has at least one coloured vertex. Then the connectivity condition (*) can be rephrased as: the set b of indices, where ψ is true, comprises all of I. So we have a distribution sequence for φ_{conn} with $\varsigma_{conn}(b) =_{def} (b = I)$ and ψ.
3. For φ_{cycle}, which says that there is a cycle in the graph, the cyclicity condition (†) can be rephrased as a distribution sequence with

$$\varsigma_{cycle}(b_1,b_2,b_3) =_{def} (b_1 \neq \emptyset)\bigvee(\exists X(X \neq \emptyset \wedge I - X \neq \emptyset \wedge b_2 = X \wedge b_3 = I - X))$$

   and ψ_1 says that there is a cycle, ψ_2 says that there are two coloured connected vertices, and ψ_3 says that there is at least one coloured vertex, and b_j $(j \in \{1,2,3\})$ is the set of indices i $(i \in I)$ of the structures, such that $\mathcal{A}_i \models \psi_j$.
4. Both considered properties, connectivity and the existence of cycles, are expressible already in FOL augmented by a unary Transitive Closure operator TC^1 or in (Least) Fixed Point Logic (LFP).

Our main theorem 7 shows that the method can be mechanized, even if (*) and (†) are not given in advance.

B Extensions of First Order Logic

In this section, we consider several extensions of First Order Logic (FOL). FOL is not powerful enough to express many useful properties. This obstacle can be overcome by adding different operators as well as by richer quantification. In our further considerations, we will need some additional logical tools and notations. For all logics we define:

Definition 15 (Quantifier Rank of Formulae)**.**
Quantifier rank of formula φ ($rank(\varphi)$) can be defined as follows:

- *for φ without quantifiers $rank(\varphi) = 0$;*
- *if $\varphi = \neg\varphi_1$ and $rank(\varphi_1) = n_1$, then $rank(\varphi) = n_1$;*
- *if $\varphi = \varphi_1 \cdot \varphi_2$, where $\cdot \in \{\vee,\wedge,\rightarrow\}$, and $rank(\varphi_1) = n_1$, $rank(\varphi_2) = n_2$, then $rank(\varphi) = max\{n_1,n_2\}$;*
- *if $\varphi = Q\varphi_1$, where Q is a quantifier, and $rank(\varphi_1) = n_1$, then $rank(\varphi) = n_1 + 1$.*

It is well known that the expressive power of FOL is very limited. For example, the transitive closure is not defined in this logic. The source of this defect is the lack of counting or recursion mechanism in this logic. Several attempts to augment the expressive power of FOL were done in this direction. For example, Immerman, cf. [21], introduced the *counting quantifier* $\exists ix$, that can be read as: "there are at lest i elements x such that ...". On the other hand, these attempts were inspired by the work by Mostowski, cf. [37], when he introduced the notion of *cardinality quantifiers* (for example: "there are infinitely many elements"), and Tarski, cf. [42], who studied the *infinitary languages*. The next development of this subject was done in the works by Linström, cf. [27, 28], which introduced *generalized quantifiers*. In this contribution, we mostly follow [24]. We use the notation $\mathcal{K}$ (or $\mathcal{Q}$) for an arbitrary class of structures. If τ is a vocabulary, $\mathcal{K}(\tau)$ is the class of structures over τ that are in $\mathcal{K}$.

Definition 16 (Simple Unary Generalized Quantifier)**.**
A simple unary generalized quantifier *is a class $\mathcal{Q}$ of structure over the vocabulary consisting of a unary relation symbol P, such that $\mathcal{Q}$ is closed under isomorphism, i.e., if $\mathcal{U} = \langle \mathcal{U}, \mathcal{P}^{\mathcal{U}} \rangle$ is a structure in $\mathcal{Q}$ and $\mathcal{U}' = \langle \mathcal{U}', \mathcal{P}^{\mathcal{U}'} \rangle$ is a structure that is isomorphic to $\mathcal{U}$, then $\mathcal{U}'$ is also in $\mathcal{Q}$.*

The *existential* quantifier is the class of all structures $\mathcal{U} = \langle \mathcal{U}, \mathcal{P}^{\mathcal{U}} \rangle$ with P^U a non-empty subset of U, while the *universal* quantifier consists of all structures of the form $\mathcal{U} = \langle \mathcal{U}, \mathcal{U} \rangle$. Numerous natural examples of simple unary generalized quantifiers on class of finite structures arise from properties that are not FOL definable on finite structures, such as "there is an even number of elements", "there are at least $log(n)$ many elements", etc. In particular, the quantifier "there is an even number of elements" can be viewed as the class: $\mathcal{Q}_{even} = \{\langle U, P^U \rangle : U \text{ is a finite set}, P^U \subseteq U, \text{ and } |P^U| \text{ is even}\}$. We may expend definition 16 to the *n-ary generalized quantifier*.

Definition 17 (Lindström Quantifiers)**.**
Let us $(n_1, n_2, ..., n_\ell)$ be a sequence of positive integers. A Lindström Quantifier of type *$(n_1, n_2, ..., n_\ell)$ is in a class $\mathcal{Q}$ of structure over the vocabulary consisting of relation symbols $(P_1, P_2, ..., P_\ell)$ such that P_i is n_i-ary for $1 \leq i \leq \ell$ and $\mathcal{Q}$ is closed under isomorphisms.*

One of the most known examples of non-simple quantifiers is the *equicardinality or Härtig quantifier I*. This is a Lindström Quantifier of type $(1, 1)$ which comprises all structures $\mathcal{U} = \langle \mathcal{U}, \mathcal{X}, \mathcal{Y} \rangle$ when $|X| = |Y|$. Another example is the *Rescher* quantifier whose mean is *more*.

Another way to extend FOL is to allow countable disjunctions and conjunctions:

Definition 18 (Infinitary Logics)**.**

- *$L_{\omega_1\omega}$ is the logic, which allows countable disjunctions and conjunctions;*
- *$L^k_{\omega_1\omega}$ is the logic, which allows countable disjunctions and conjunctions, but has only a total of k distinct variables;*
- *$L^k_{\infty\omega}, k \geq 1$, is the logic, which allows infinite disjunctions and conjunctions, but has only a total of k distinct variables;*
- $L^\omega_{\infty\omega} = \bigcup L^k_{\infty\omega}$.

We assume that only variables involved , are $v_0, \ldots, v_{k-1}$.

Now, we introduce the syntax and the semantics of the logic $L^k_{\infty\omega}$ that contains simple unary generalized quantifiers.

Definition 19.
Let $\mathbf{Q} = \{Q_i : i \in I\}$ be a family of simple unary generalized quantifiers and let k be a positive integer. The infinitary logic $L^k_{\infty\omega}(\mathbf{Q})$ with k variables and the generalized quantifiers $\mathbf{Q}$ has the following syntax (for any vocabulary τ):

- *the variables of* $L^k_{\infty\omega}(\mathbf{Q})$ *are* $v_1, ..., v_k$*;*
- $L^k_{\infty\omega}(\mathbf{Q})$*contains all FOL formulae over* τ *with variables among* $v_1, ..., v_k$*;*
- *if* φ *is a formulae of* $L^k_{\infty\omega}(\mathbf{Q})$*, then so is* $\neg\varphi$*;*
- *if* Ψ *is a set of formulae of* $L^k_{\infty\omega}(\mathbf{Q})$*, then* $\bigvee\Psi$ *and* $\bigwedge\Psi$ *are also formulae of* $L^k_{\infty\omega}(\mathbf{Q})$*;*
- *if* φ *is a formulae of* $L^k_{\infty\omega}(\mathbf{Q})$*, then each of the expressions* $\exists v_j\varphi, \forall v_j\varphi, Q_i v_j\varphi$ *is also a formulae of* $L^k_{\infty\omega}(\mathbf{Q})$ *for every* j *such that* $1 \le j \le k$ *and for every* $i \in I$*.*

The semantic of $L^k_{\infty\omega}(\mathbf{Q})$ is defined by induction on the construction of the formulae. So, $\bigvee\Psi$ is interpreted as a disjunction over all formulae in Ψ and $\bigwedge\Psi$ is interpreted as a conjunction. Finally, if $\mathcal{U}$ is the structure having U as its universe and $\varphi(v_j, \bar{y})$ is a formulae of $L^k_{\infty\omega}(\mathbf{Q})$ with free variables among the variables of v_j and the variables in the sequence $\bar{y}$, and $\bar{u}$ is a sequence of elements from the universe of $\mathcal{U}$, then: $U, \bar{u} \models Q_i v_j\varphi(v_j, \bar{y})$ iff the structure $\langle U, \{a : U, a, \bar{u} \models \varphi(v_j, \bar{y})\}\rangle$ is in the quantifier $\mathcal{Q}_i$.

We may also enrich the expressive power of FOL by allowing quantification over relation symbols. Second Order Logic (SOL) is like FOL, but allows also variables and quantification over relation variables of various but fixed arities. Monadic Second Order Logic ($MSOL$) is the sublogic of SOL where relation variables are restricted to be unary. The meaning function of formulae is explained for arbitrary τ–structures, where τ is the vocabulary, i.e., a finite set of relation and constant symbols. Fixed Point Logic (LFP) can be viewed as a fragment of SOL, where the second order variables only occur positively and in the fixed point construction. Similarly $MLFP$ corresponds to the case where the arity of the relation variables is restricted to 1. The semantics of the fixed point is given by the least fixed point, which does always exist because of the positivity assumption on the set variable. The Logic LFP is defined similarly with operators k-LFP for every $k \in N$ which bind $2k$ variables. On ordered structures LFP expresses exactly the polynomially recognizable classes of finite structures. Without order, every formula in LFP has a polynomial model checker. For transition systems, $MLFP$ corresponds exactly to μ-calculus, cf. [43].

The logic MTC (Monadic Transitive Closure) is defined inductively, like FOL. For a thorough discussion of it, cf. [22]. Atomic formulae are as usual. The inductive clauses include closure under the boolean operations, existential and universal quantification and one more clause: If $\phi(x, y, \bar{u})$ is a MTC–formula with x, y and $\bar{u} = u_1, \ldots, u_n$ its free variables, s, t are terms, then $MTCx, y, s, t\phi(x, y, \bar{u})$ is a MTC–formula with x, y bound and $\bar{u}$ free. The formula $MTCx, y, s, t\phi(x, y, \bar{u})$ holds in a structure $\mathcal{U}$ under an assignment of variables z if $s_z, t_z \in TrCl(\phi^{\mathcal{U}})$. The logic TC is defined similarly with operators k-TC for every $k \in N$ which bind $2k$ variables. For more detailed exposition, cf. [12, 17].

In [17], E. Grädel introduced a generalization of Ehrenfeucht–Fraïssé Games for TC. As we need this game in the proof of Theorem 2, we give Grädel's definition and Theorem 9 concerning Pebble Games for TC verbatim as in [17].

Definition 20 (Ehrenfeucht–Fraïssé Games for TC)**.**
Suppose we have two structures $\mathcal{U}$ *and* $\mathcal{V}$ *of the same vocabulary* σ*. Let* $c_1, \ldots, c_s$ *and* $d_1, \ldots, d_s$ *be the interpretation of the constants of* σ *in* $\mathcal{U}$ *and* $\mathcal{V}$*, respectively. The* k*–pebble game on the pair* $(\mathcal{U}, \mathcal{V})$ *is played by Players I and II as follows: There are* k *pairs* $(u_1, v_1), \ldots, (u_k, v_k)$ *of pebbles. Each round of the game consists of either an* $\exists$*–move,* $\forall$*–move or* TC*–move:*
$\exists$**–move:** Player I places a yet unused pebbles u_i on an element of $\mathcal{U}$. Player II answers by putting the corresponding pebble v_i on $\mathcal{V}$.
$\forall$**–move:** Similarly but with "reversed board": Player I places v_i on $\mathcal{V}$. Player II responds with u_i on $\mathcal{U}$.

TC–**move:** Suppose that r pairs of pebbles are already on the board. For some $l \leq (k-r)/2$, Player I selects a sequence $\bar{x}_0, \ldots, \bar{x}_m$ of l–tuples in $\mathcal{U}$ such that $\bar{x}_0$ and $\bar{x}_m$ consist only of sets of constants and already pebbled elements. Player II indicates a similar sequences (not necessary of the same length) of l–tuples $\bar{y}_0, \ldots, \bar{y}_n$ in $\mathcal{V}$ where $\bar{y}_0 = f(\bar{x}_0), \ldots, \bar{y}_n = f(\bar{x}_m)$. Player I then selects some $i \leq n$ and places $2l$ (yet unused) pebbles on $\bar{y}_i$ and $\bar{y}_{i+1}$. Player II selects some a $j \leq m$ and places the corresponding pebbles on $\bar{x}_j$ and $\bar{x}_{j+1}$.
$\neg TC$–**move:** is like *TC*–move, but with structures $\mathcal{A}$ and $\mathcal{V}$ interchanged.
When all pebbles are placed, Player I wins if the pebbles determine a local isomorphism from $\mathcal{U}$ to $\mathcal{V}$. More precisely: Let $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$ be the elements carrying the pebbles $u_1, \ldots, u_k$ and $v_1, \ldots, v_k$. If the mapping f with $f(a_i) = b_i$ for $i = 1, \ldots, k$, and $f(c_i) = d_i$ for $i = 1, \ldots, s$, is an isomorphism between the substructures of $\mathcal{A}$ and $\mathcal{V}$ that are generated by the pebbles elements and the constants, then Player II wins; otherwise, Player I wins.

Theorem 9 (Grädel, [17])**.**
For all structures $\mathcal{U}$ and $\mathcal{V}$ and all $k \in N$, the following are equivalent: Player II has a winning strategy for the TC–game with k pebbles on $(\mathcal{U}, \mathcal{V})$ **and** $\mathcal{U} \equiv^k_{TC} \mathcal{V}$**.**

B.1 Complexity of Computation for Extensions of First Order Logic

Computation for *FOL* is polynomial (even in logarithmic space), whereas computation for *MSOL* is likely to be non–polynomial, as it sits fully in the polynomial hierarchy.

More precisely, the complexity of computation (in the size of the structure) of Second Order Logic expressible properties can be described as follows. The class *NP* of non-deterministic polynomial-time problems is the set of properties, which are expressible by Existential Second Order Logic on finite structures, cf. [13]. Computation for *SOL* definable properties is in the polynomial hierarchy, cf. [16]. Moreover, for every level of the polynomial hierarchy there is a problem, expressible in *SOL*, that belongs to this class. The same fact hold for *MSOL*, too, as observed in [30].

Computation for properties, definable in Fixed Point Logic, is polynomial, cf. [43]. The relation between *FOL* with generalized quantifiers and computations with oracles is investigated in [29]. Most properties, which appears in real life applications, are stronger than *FOL* but weaker than *MSOL*, and their computational complexity is polynomial.

C Weighted Monadic Second Order Logic and Weighted Tree Automata

In this section, we follow [9] almost verbatim. Let $\mathbf{N} = \{1, 2, \ldots\}$ be the set of natural numbers and let $\mathbf{N}_0 = \mathbf{N} \cup \{0\}$. A ranked alphabet is a pair $(\Sigma, \mathbf{rk}_\Sigma)$ consisting of a finite alphabet Σ and a mapping $\mathbf{rk}_\Sigma : \Sigma \to \mathbf{N}_0$, which assigns to each symbol of Σ its rank. By $\Sigma^{(m)}$ we denote the set of all symbols with rank $m \in \mathbf{N}_0$ and $a^{(m)}$ denotes that $a \in \Sigma^{(m)}$. Let $\max_\Sigma = \max\{\mathbf{rk}_\Sigma(a) \mid a \in \Sigma\}$, the maximal rank of Σ. Let $\mathbf{N}^*$ be the set of all finite words over $\mathbf{N}$. A *tree domain* $\mathcal{B}$ is a finite, non-empty subset of $\mathbf{N}^*$ such that for all $u \in \mathbf{N}^*$ and $i \in \mathbf{N}$, $u.i$ is the prefix of u of length i. Moreover, $u.i \in \mathcal{B}$ implies $u.1, \ldots, u.(i-1) \in \mathcal{B}$, $u.1, \ldots, u.(i-1)$ is called *immediate prefix* of $u.i$; the immediate prefix of $u.1$ is the empty set ϵ. Note that the tree domain of $\mathcal{B}$ is prefix-closed. A *tree over a set L (of labels)* is a mapping $t : \mathcal{B} \to L$, such that $dom(t) = \mathcal{B}$ is a tree domain, $im(t)$ is the image of t. The elements of $dom(t)$ are called *positions* of t and $t(u)$ is called *label* of t at $u \in dom(t)$. The set of all trees over L is denoted by T_L.

Definition 21 (Tree Valuation Monoid)**.**
A tv-monoid is a quadruple $\mathcal{D} = (D, +, Val, \mathbf{0})$ *such that* $(D, +, \mathbf{0})$ *is a commutative monoid and* $Val : T_D \to D$ *is a function with* $Val(d) = d$ *for every tree* $d \in T_D$ *and* $Val(t) = \mathbf{0}$*, whenever* $\mathbf{0} \in im(t)$ *for* $t \in T_D$*.*

Val is called a *(tree) valuation function.*

Definition 22 (Product Tree Valuation Monoid)**.**
A ptv-monoid $\mathcal{D} = (D, +, Val, \diamondsuit, \mathbf{0}, \mathbf{1})$ *consists of a tree valuation monoid, a constant* $\mathbf{1} \in D$ *with* $Val(t) = \mathbf{1}$*, whenever* $im(t) = \{\mathbf{1}\}$ *for* $t \in T_D$*, and an operation* $\diamondsuit : D^2 \to D$ *with* $\mathbf{0} \diamondsuit d = d \diamondsuit \mathbf{0} = \mathbf{0}$ *and* $\mathbf{1} \diamondsuit d = d \diamondsuit \mathbf{1} = \mathbf{1}$*.*

Note that the operation $\diamondsuit$, in general, has to be neither commutative nor associative.

C.1 Weighted Monadic Second Order Logic

Given a ptv-monoid D, the syntax of $WMSOL$ over D is defined by the following way:

Boolean formulae:

- $label_a(x)$ and $edge_i(x, y)$ for $a \in \Sigma$ and $1 \leq i \leq \max_\Sigma$;
- $x \in X$, $\neg\beta_1$, $\beta_1 \wedge \beta_2$, $\forall x \beta_1$, $\forall X \beta_1$ for first order variable x and second order variable X.

Weighted formulae:

- d for $d \in D$;
- β for boolean formula β;
- $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, $\exists x \phi_1$, $\forall x \phi_1$, $\exists X \phi_1$, $\forall X \phi_1$.

The set $free(\phi)$ of free variables occurring in ϕ is defined as usual. Semantics of $WMSOL$ valuates trees by elements of D. There is no change in semantics of boolean formulae. $\mathbf{0}$ defines the semantics of the truth value "false". $\mathbf{1}$ defines the semantics of the truth value "true". The monoid operation "+" is used to define semantics of disjunction and existential quantifier. The monoid Val function is used to define the semantics of the first order universal quantification. If, for example, we use the max-plus-semiring the semantical interpretation of $\forall x \phi$ is the sum of all weights (rewards or time) defined by ϕ for all different positions x. More precisely, for a $(\mathcal{V}, t)$-assignment that maps $\tilde{\sigma} : \mathcal{V} \to dom(t) \cup PS(dom(t))$, with $\tilde{\sigma}(x) \in dom(t)$ and $\tilde{\sigma}(X) \subseteq dom(t)$, and $s \in T_{\Sigma_\mathcal{V}}$. The formal definition of the semantics of $WMSOL$ can be seen in [9].

C.2 Expressive Power of Weighted Monadic Second Order Logic

$WMSOL$ and its fragments have a considerable expressive power. In [35], the coincidence of recognizable trace series with those, which are definable by restricted formulae from a weighted logics over traces, was proved. In [15], a notion of a $WMSOL$ logics over pictures was introduced, weighted 2-dimensional on-line tessellation automata ($W2OTA$) was defined and it was proved that for commutative semirings, the class of picture series defined by sentences of the weighted logics coincides with the family of picture series that are computable by $W2OTA$. In [33], quantitative models for texts were investigated, an algebraic notion of recognizability was defined and it was shown that recognizable text series coincide with text series definable in weighted logics. Nested words are a model for recursive programs. In [34], quantitative extensions of nested word series were considered and it was shown that regular nested word

series coincide with series definable in weighted logics. Moreover, lots of optimization problems and counting problems are expressible in $WMSOL$, cf. [8, 11, 32, 1, 10]. The logic may be used in order to describe data mining problems, decision making, planning and scheduling.

Additionally, in [9], a strong relationship between WTA over ptv-monoids and the fragments of the $WMSOL$ was established. Let Σ be a ranked alphabet and $(D, +, Val, \mathbf{0})$ a tv-monoid.

Definition 23 (Weighted Bottom-up Tree Automaton)**.**
A WTA over a tv-monoid D is a quadruple $M = \langle Q; \Sigma, \mu, F\rangle$, where Q is a non-empty finite set of states, Σ is a ranked alphabet, $\mu = (\mu_m)_{0\le m\le \max\Sigma}$ is a family of transition mappings $\mu_m : \Sigma^{(m)} \to D^{Q^m\times Q}$, and $F \subseteq Q$ is a set of final states.

However, the larger the particular fragment gets, the more restrictions on the underlying ptv-monoid we need. The main theorem of [9] states (cf. the exact definitions in the paper):

Theorem 10. *Let $S : T_\Sigma \to D$ be a tree series.*

1. *If D is* **regular***, then S is recognizable iff S is definable by a $\forall$-restricted and* **strongly** *$\wedge$-restricted $WMSOL$ sentence ϕ.*
2. *If D is* **left-distributive***, then S is recognizable iff S is definable by a $\forall$-restricted and $\wedge$-restricted $WMSOL$ sentence ϕ.*
3. *If D is a cctv-***semiring***, then S is recognizable iff S is definable by a $\forall$-restricted and* **commutatively** *$\wedge$-restricted $WMSOL$ sentence ϕ.*

D Proof of Theorem 2 for MTC^m

In this part of the Appendices, we give our original proof of the following part of Theorem 2:

Theorem 11. *Let $\mathcal{I}$ be an index structure. Then $DJ - PP(MTC^m, MTC^m)$ and $FShu - PP(MTC^m, MTC^m)$ hold.*

Proof: *We use pebble game for MTC as introduced in [17].*
$\exists$**–move:** *If Player I puts pebble u on some element a of structure $\mathcal{A}_i$ for some $i \in I$, Player II now places her pebble v on b of structure $\mathcal{B}_i$ using the winning strategy of the components.*
$\forall$**–move** *is proved similarly.*
MTC**–move:** *If Player I selects a sequence $x_0,\ldots,x_m$ in $\mathcal{A}$, we divide the sequence into segments $x_{m_{0,0}},\ldots,x_{m_{0,1}},x_{m_{1,0}},\ldots,x_{m_{1,1}}, \ldots,x_{m_{p,0}},\ldots,x_{m_{p,1}}$ with $m_{0,0} = 0$, $m_{p,1} = m$ and such that each subsequence $x_{m_{q,0}},\ldots,x_{m_{q,1}}$ lies in the same component $\mathcal{A}_{i_q}$.*
Player II now constructs her sequence $y_0,\ldots,y_n$ in $\mathcal{B}$ segment-wise as follows: She uses two auxiliary pebbles U_0, U_1 and V_0, V_1 on each structure. For the segment $x_{m_{q,0}},\ldots,x_{m_{q,1}}$ she puts U_0 on $x_{m_{q,0}}$ and U_1 on $x_{m_{q,1}}$. Using the winning strategy on components i_q she places the pebbles V_0 and V_1 on elements $y_{m_{q,0}}$ and $y_{m_{q,1}}$ and chooses the intermediate elements according to the winning strategy of the MTC–move. The auxiliary pebbles are reused after every segment. If Player I now pebbles two neighboring elements in the sequence $y_0,\ldots,y_n$ in $\mathcal{B}$, two cases can occur:

1. *If both pebbled elements are in the same component, Player II just follows her winning strategy on the corresponding component on $\mathcal{A}$.*
2. *If the two pebbled elements are in different components, then she plays accordingly.*

It is now easy to verify that this is indeed a winning strategy. The auxiliary pebbles are only used temporarily to mark the beginning and the end of the segments.

Note that the index set is part of the structures $\mathcal{A}$ and $\mathcal{B}$, and is treated itself like a component. However, in this component Player II copies faithfully the moves of Player I.

Acknowledgments

We would like to thank Prof. J.A. Makowsky for valuable discussions.