Expressivity Within Second-Order Transitive-Closure Logic

Flavio Ferrarotti
Software Competence Center Hagenberg, Hagenberg, Austria
flavio.ferrarotti@scch.at
🆔 https://orcid.org/0000-0003-2278-8233

Jan Van den Bussche
Hasselt University, Hasselt, Belgium
jan.vandenbussche@uhasselt.be
🆔 https://orcid.org/0000-0003-0072-3252

Jonni Virtema
Hasselt University, Hasselt, Belgium
jonni.virtema@uhasselt.be
🆔 https://orcid.org/0000-0002-1582-3718

Abstract

Second-order transitive-closure logic, SO(TC), is an expressive declarative language that captures the complexity class PSPACE. Already its monadic fragment, MSO(TC), allows the expression of various NP-hard and even PSPACE-hard problems in a natural and elegant manner. As SO(TC) offers an attractive framework for expressing properties in terms of declaratively specified computations, it is interesting to understand the expressivity of different features of the language. This paper focuses on the fragment MSO(TC), as well on the purely existential fragment SO(2TC)($\exists$); in 2TC, the TC operator binds only tuples of relation variables. We establish that, with respect to expressive power, SO(2TC)($\exists$) collapses to existential first-order logic. In addition we study the relationship of MSO(TC) to an extension of MSO(TC) with counting features (CMSO(TC)) as well as to order-invariant MSO. We show that the expressive powers of CMSO(TC) and MSO(TC) coincide. Moreover we establish that, over unary vocabularies, MSO(TC) strictly subsumes order-invariant MSO.

1 Introduction

Second-order transitive-closure logic, SO(TC), is an expressive declarative language that captures the complexity class PSPACE [21]. It extends second-order logic with a transitive closure operator over relations of relations, i.e., over super relations among relational structures. The super relations are defined by means of second-order logic formulae with free relation variables. Already its monadic fragment, MSO(TC), allows the expression of NP-complete problems in a natural and elegant manner. Consider, for instance, the well known Hamiltonian cycle query over the standard vocabulary of graphs, which is not expressible in monadic second-order logic [13].

▶ **Example 1.** A graph $G = (V, E)$ has a Hamiltonian cycle if the following holds:

a. There is a relation $\mathcal{R}$ such that $(Z, z, Z', z') \in \mathcal{R}$ iff $Z' = Z \cup \{z'\}$, $z' \notin Z$, and $(z, z') \in E$.

b. The tuple $(\{x\}, x, V, y)$ is in the transitive closure of $\mathcal{R}$, for some $x, y \in V$ s.t. $(y, x) \in E$.

In the language of MSO(TC) this can be written as follows:

$$\exists XYxy\big(X(x) \wedge \forall z(z \neq x \rightarrow \neg X(x)) \wedge \forall z(Y(z)) \wedge E(y, x) \wedge [\text{TC}_{Z,z,Z',z'}\varphi](X, x, Y, y)\big),$$

where $\varphi := \neg Z(z') \wedge \forall x\big(Z'(x) \leftrightarrow (Z(x) \vee z' = x)\big) \wedge E(z, z')$.

Even some well-known PSPACE-complete problems such as deciding whether a given quantified Boolean formula QBF is valid [27] can be expressed in MSO(TC) (see Section 3).

In general, SO(TC) offers an attractive framework for expressing properties in terms of declaratively specified computations at a high level of abstraction. There are many examples of graph computation problems that involve complex conditions such as graph colouring [4], topological subgraph discovery [19], recognition of hypercube graphs [18], and many others (see [9, 16, 17]). Such graph algorithms are difficult to specify, even by means of rigorous methods such as Abstract State Machines (ASMs) [10], B [2] or Event-B [3], because the algorithms require the definition of characterising conditions for particular subgraphs that lead to expressions beyond first-order logic. Therefore, for the sake of easily comprehensible and at the same time fully formal high-level specifications, it is reasonable to explore languages such as SO(TC). Let us see an example that further supports these observations.

▶ **Example 2.** Self-similarity of complex networks [37] (aka scale invariance) has practical applications in diverse areas such as the world-wide web [14], social networks [20], and biological networks [32]. Given a network represented as a finite graph G, it is relevant to determine whether G can be built starting from some graph pattern G_p by recursively replacing nodes in the pattern by new, "smaller scale", copies of G_b. If this holds, then we say that G is self-similar.

Formally, a graph G is *self-similar* w.r.t. a graph pattern G_p of size k, if there is a sequence of graphs $G_0, G_1, \ldots, G_n$ such that $G_0 = G_p$, $G_n = G$ and, for every pair (G_i, G_{i+1}) of consecutive graphs in the sequence, there is a partition $\{P_1, \ldots, P_k\}$ of the set of nodes of G_{i+1} which satisfies the following:

a. For every $j = 1, \ldots, k$, the sub-graph induced by P_j in G_{i+1} is isomorphic to G_i.

b. There is a graph G_t isomorphic to G_p with set of nodes $V_t = \{a_1, \ldots, a_k\}$ for some $a_1 \in P_1, \ldots, a_k \in P_k$ and set of edges

$$E_t = \{(a_i, a_j) \mid \text{there is an edge } (x, y) \text{ of } G_{i+1} \text{ such that } P_i(x) \text{ and } P_j(y)\}.$$

c. For very $1 \leq i < j \leq k$, the closed neighborhoods $N_{G_{i+1}}[P_i]$ and $N_{G_{i+1}}[P_j]$ of P_i and P_j in G_{i+1}, respectively, are isomorphic.

It is straightforward to write this definition of self-similarity in SO(TC), for we can clearly write a second-order logic formula which defines such a super relation $\mathcal{R}$ on graphs and then simply check whether the pair of graphs (G, G_p) is in the transitive closure of $\mathcal{R}$.

Highly expressive query languages are gaining relevance in areas such as knowledge representation (KR), rigorous methods and provers. There are several examples of highly expressive query languages related to applications in KR. See for instance the monadically defined queries in [36], the Monadic Disjunctive SNP queries in [5] or the guarded queries in [11]. See also [33] where a query language with transitive closure for graph databases is considered. All of them can be considered fragments of Datalog. Regarding rigorous methods, the TLA$^+$ language [28] is able to deal with higher-order formulations, and tools such as the

TLA$^+$ Proof System[1] and the TLA$^+$ Model-Checker (TLC)[2] can handle them (provided a finite universe of values for TLC). Provers such as Coq[3] and Isabelle[4] can already handle some high-order expression. Moreover, the success with solvers for the Boolean satisfiability problem (SAT) has encouraged researchers to target larger classes of problems, including PSPACE-complete problems, such as satisfiability of Quantified Boolean formulas (QBF). Note the competitive evaluations of QBF solvers (QBFEVAL) held in 2016 and 2017 and recent publications on QBF solvers such as [8, 31, 22] among several others.

We thus think it is timely to study which features of highly expressive query languages affect their expressive power. In this sense, SO(TC) provides a good theoretical base since, apart from been a highly expressive query language (recall that it captures PSPACE), it enables natural and precise high-level definitions of complex practical problems, mainly due to its ability to express properties in terms of declaratively specified computations. While second-order logic extended with the standard partial fixed-point operator, as well as first-order logic closed under taking partial fixed-points and under an operator for non-deterministic choice, also capture the class of PSPACE queries over arbitrary finite structure [34], relevant computation problems such as that in Example 2 are clearly more difficult to specify in these logics. The same applies to the extension of first-order logic with the partial fixed-point operator, which is furthermore subsumed by SO(TC) since it captures PSPACE only on the class of ordered finite structures [1]. Note that SO(TC) coupled with hereditary finite sets and set terms, could be considered as a kind of declarative version of Blass, Gurevich, and Shelah (BGS) model of abstract state machine [7], which is a powerful language in which all computable queries to relational databases can be expressed [6].

Our results can be summarized as follows.

1. We investigate to what extent universal quantification and negation are important to the expressive power of SO(TC). Specifically, we consider the case where TC-operators are applied only to second-order variables. Of course, a second-order variable can simulate a first-order variable, since we can express already in first-order logic (FO) that a set is a singleton. This, however, requires universal quantification.

   We define a "purely existential" fragment of SO(TC), SO(2TC)($\exists$), as the fragment without universal quantifiers and in which TC-operators occur only positively and bind only tuples of relation variables. We show that the expressive power of this fragment collapses to that of existential FO.

   For SO alone, this collapse is rather obvious and was already remarked by Rosen in the introduction of his paper [35]. Our result generalizes this collapse to include TC operators, where it is no longer obvious.

2. We investigate the expressive power of the monadic fragment, MSO(TC). On strings, this logic is equivalent to the complexity class NLIN. Already on unordered structures, however, we show that MSO(TC) can express counting terms and numeric predicates in NLOGSPACE. In particular, MSO(TC) can express queries not expressible in the fixpoint logic FO(LFP). We also discuss the fascinating open question whether the converse holds as well.

3. We compare the expressive power of MSO(TC) to that of order-invariant MSO. Specifically, we show that MSO(TC) can express queries not expressible in order-invariant MSO; over monadic vocabularies, we show that order-invariant MSO is subsumed by MSO(TC). Again, what happens over higher-arity relations is an interesting open question.

---

[1] `https://tla.msr-inria.inria.fr/tlaps`
[2] `https://lamport.azurewebsites.net/tla/tlc.html`
[3] `https://coq.inria.fr/`
[4] `https://isabelle.in.tum.de/`

This paper is organized as follows. In Section 2 definitions and basic notions related to SO(TC) are given. In Section 3 the complexity of model checking is studied. Section 4 is dedicated to establishing the collapse of SO(2TC)(∃) to existential first-order logic. Sections 5 and 6 concentrate on the relationships between MSO(TC) and the counting extension CMSO(TC) and order-invariant MSO, respectively. We conclude with a discussion of open questions in Section 7.

2   Preliminaries

We assume that the reader is familiar with finite model theory, see e.g., [15] for a good reference. For a tuple $\vec{a}$ of elements, we denote by $\vec{a}[i]$ the ith element of the tuple. We recall from the literature, the syntax and semantics of first-order (FO) and second-order (SO) logic, as well as their extensions with the transitive closure operator (TC). We assume a sufficient supply of *first-order* and *second-order variables*. The natural number $\mathrm{ar}(R) \in \mathbb{N}$, is the *arity* of the second-order variable X. By *variable*, we mean either a first-order or second-order variable. Variables χ and χ' have the same *sort* if either both χ and χ' are first-order variables, or both are second-order variables of the same arity. Tuples $\vec{\chi}$ and $\vec{\chi'}$ of variables have the same *sort*, if the lengths of $\vec{\chi}$ and $\vec{\chi'}$ are the same and, for each i, the sort of $\vec{\chi}[i]$ is the same as the sort of $\vec{\chi}[i]$.

▶ **Definition 3.** The formulas of SO(TC) are defined by the following grammar:

$$\varphi ::= x = y \mid X(x_1, \ldots, x_k) \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists x \varphi \mid \exists Y \varphi \mid [\mathrm{TC}_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'}),$$

where X and Y are second-order variables, $k = \mathrm{ar}(X)$, $x, y, x_1, \ldots, x_k$ are first-order variables, $\vec{X}$ and $\vec{X'}$ are disjoint tuples of variables of the same sort, and $\vec{Y}$ and $\vec{Y'}$ are also tuples of variables of that same sort (but not necessarily disjoint).

The set of free variables of a formula φ, denoted by $\mathrm{FV}(\varphi)$ is defined as usual. For the TC operator, we define

$$\mathrm{FV}([\mathrm{TC}_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'})) := (\mathrm{FV}(\varphi) - (\vec{X} \cup \vec{X'})) \cup \vec{Y} \cup \vec{Y'}.$$

Above in the right side, in order to avoid cumbersome notation, we use $\vec{X}$, $\vec{X'}$, $\vec{Y}$ and $\vec{Y'}$ to denote the sets of variables occurring in the tuples.

   A *vocabulary* is a finite set of variables. A (finite) *structure* $\mathfrak{A}$ over a vocabulary τ is a pair (A, I), where A is a finite nonempty set called the *domain* of $\mathfrak{A}$, and I is an *interpretation* of τ on A. By this we mean that whenever $x \in \tau$ is a first-order variable, then $I(x) \in A$, and whenever $X \in \tau$ is a second-order variable of arity m, then $I(X) \subseteq A^m$. In this article, structures are always finite. We denote $I(X)$ also by $X^{\mathfrak{A}}$. For a variable X and a suitable value R for that variable, $\mathfrak{A}[R/X]$ denotes the structure over $\tau \cup \{X\}$ equal to $\mathfrak{A}$ except that X is mapped to R. We extend the notation also to tuples of variables and values, $\mathfrak{A}[\vec{X}/\vec{R}]$, in the obvious manner. We say that a vocabulary τ is *appropriate* for a formula φ if $\mathrm{FV}(\varphi) \subseteq \tau$.

▶ **Definition 4.** Let $\mathfrak{A}$ be a structure over τ and φ an SO(TC)-formula such that τ is appropriate for φ. The satisfaction of φ by $\mathfrak{A}$, denoted by $\mathfrak{A} \models \varphi$, is defined as follows. We only give the cases for second-order quantifiers and transitive closure operator; the remaining cases are defined as usual.

   ▬ For second-order variable X: $\mathfrak{A} \models \exists X \varphi$ iff $\mathfrak{A}[R/X] \models \varphi$, for some $R \subseteq A^{\mathrm{ar}(X)}$.

▬ For the case of the TC-operator, consider a formula ψ of the form $[\mathrm{TC}_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'})$ and let $\mathfrak{A} = (A, I)$. Define $\mathcal{J}_{\vec{X}}$ to be the following set

$$\{J(\vec{X}) \mid J \text{ is an interpretation of } \vec{X} \text{ on } A\} = \{J(\vec{X'}) \mid J \text{ is an interpretation of } \vec{X'} \text{ on } A\}$$

and consider the binary relation $\mathcal{B}$ on $\mathcal{J}_{\vec{X}}$ defined as follows:

$$\mathcal{B} := \{(\vec{R}, \vec{R'}) \in \mathcal{J}_{\vec{X}} \times \mathcal{J}_{\vec{X}} \mid \mathfrak{A}[\vec{R}/\vec{X}, \vec{R'}/\vec{X'}] \models \varphi\}.$$

We set $\mathfrak{A} \models \psi$ to hold if $(I(\vec{Y}), I(\vec{Y'}))$ belongs to the transitive closure of $\mathcal{B}$. Recall that, for a binary relation $\mathcal{B}$ on any set $\mathcal{J}$, the transitive closure of $\mathcal{B}$ is defined by

$$\mathrm{TC}(\mathcal{B}) := \{(a, b) \in \mathcal{J} \times \mathcal{J} \mid \exists n > 0 \text{ and } e_0, \ldots, e_n \in \mathcal{J}$$
$$\text{such that } a = e_0,\ b = e_n, \text{ and } (e_i, e_{i+1}) \in \mathcal{B} \text{ for all } i < n\}.$$

By TC^m we denote the variant of TC in which the quantification of n above is restricted to natural numbers $\leq m$. That is, $\mathrm{TC}^m(\mathcal{B})$ consists of pairs $(\vec{a}, \vec{b})$ such that $\vec{b}$ is reachable from $\vec{a}$ by $\mathcal{B}$ in at most m steps. Moreover, by 2TC and $2\mathrm{TC}^m$ we denote the syntactic restrictions of TC and TC^m of the form

$$[\mathrm{TC}_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'}) \text{ and } [\mathrm{TC}^m_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'}),$$

where $\vec{X}$, $\vec{X'}$, $\vec{Y}$, $\vec{Y'}$ are tuples of second-order variables (i.e. without first-order variables). The logic SO(2TC) then denotes the extension of second-order logic with 2TC-operator. Analogously, by FO(1TC), we denote the extension of first-order logic with applications of such transitive-closure operators that bind only first-order variables.[5]

3 Complexity of MSO(TC)

The descriptive complexity of different logics with the transitive closure operator has been thoroughly studied by Immerman. Let SO(arity k)(TC) denote the fragment of SO(TC) in which second-order variables are all of arity $\leq k$.

▶ **Theorem 5** ([23, 24]).
▬ *On finite ordered structures, first-order transitive-closure logic* FO(1TC) *captures non-deterministic logarithmic space* NLOGSPACE.
▬ *On strings (word structures),* SO(arity k)(TC) *captures the complexity class* NSPACE(n^k).
See also the discussion in the conclusion section.

By the above theorem, MSO(TC) captures nondeterministic linear space NLIN over strings. Deciding whether a given *quantified Boolean formula* is valid (QBF) is a well-known PSPACE-complete problem [27]. Observe that there are PSPACE-complete problems already in NLIN; in fact QBF is such a problem. Thus, we can conclude the following. The inclusion in PSPACE is clear.

▶ **Proposition 6.** *Data complexity of model checking of* MSO(TC) *is* PSPACE-*complete.*

We next turn to combined complexity of model checking. By the above proposition, this is at least PSPACE-hard. However, the straightforward algorithm for model checking MSO(TC) clearly has polynomial-space combined complexity. We thus conclude:

---

[5] In the literature FO(1TC) is often denoted by FO(TC).

▶ **Proposition 7.** *Combined complexity of model checking of* MSO(TC) *is* PSPACE-*complete.*

For combined complexity, we can actually sharpen the PSPACE-hardness; already a very simple fragment of MSO(TC) is PSPACE-complete.

Specifically, we give a reduction from the *corridor tiling* problem, which is a well-known PSPACE-complete problem. Instance of the corridor tiling problem is a tuple $P = (T, H, V, \vec{b}, \vec{t}, n)$, where $n \in \mathbb{N}$ is a positive natural number, $T = \{1, \ldots, k\}$, for some $k \in \mathbb{N}$, is a finite set of *tiles*, $H, V \subseteq T \times T$ are *horizontal* and *vertical constraints*, and $\vec{b}, \vec{t}$ are n-tuples of tiles from T. A *corridor tiling for P* is a function $f : \{1, \ldots, n\} \times \{1 \ldots, m\} \to T$, for some $m \in \mathbb{N}$, such that

- $\big(f(1,1), \ldots f(n,1)\big) = \vec{b}$ and $\big(f(1,m), \ldots f(n,m)\big) = \vec{t}$,
- $\big(f(i,j), f(i+i,j)\big) \in H$, for $i < n$ and $j \le m$,
- $\big(f(i,j), f(i,j+1)\big) \in V$, for $i \le n$ and $j < m$.

The *corridor tiling problem* is the following PSPACE-complete decision problem [12]:
Input: An instance $P = (T, H, V, \vec{b}, \vec{t}, n)$ of the corridor tiling problem.
Output: Does there exist a corridor tiling for P?

Let *monadic* 2TC[∀FO] denote the fragment of MSO(2TC) of the form $[\mathrm{TC}_{\vec{X}, \vec{X'}}\varphi](\vec{Y}, \vec{Y'})$, where φ is a formula of universal first-order logic (i.e., φ is of the form $\forall \vec{x}\,\psi$, where ψ is a quantifier-free formula of first-order logic).

▶ **Theorem 8.** *Combined complexity of model checking for monadic* 2TC[∀FO] *is* PSPACE-*complete.*

Proof. Inclusion to PSPACE follows from the corresponding result for MSO(TC). In order to prove hardness, we give a reduction from corridor tiling. Let $P = (T, H, V, \vec{b}, \vec{t}, n)$ be an instance of the corridor tiling problem and set $k := |T|$. Let $\tau = \{s, X_1, \ldots X_k, Y_1, \ldots, Y_k\}$ be a vocabulary, where s is a binary second-order variable and $X_1, \ldots X_k, Y_1, \ldots, Y_k$ are monadic second-order variables. Let $\mathfrak{A}_P$ denote the structure over τ such that $A = \{1, \ldots, n\}$, $I(s)$ is the canonical successor relation on A, and, for each $i \le k$, $I(X_i) = \{j \in A \mid \vec{b}[j] = i\}$ and $I(Y_i) = \{j \in A \mid \vec{t}[j] = i\}$. Define

$$\varphi_H := \forall xy\big(s(x,y) \to \bigvee_{(i,j) \in H} Z_i'(x) \wedge Z_j'(y)\big), \quad \varphi_V := \forall x \bigvee_{(i,j) \in V} Z_i(x) \wedge Z_j'(x)$$

$$\varphi_T := \forall x \bigvee_{i \in T} \big(Z_i'(x) \wedge \bigwedge_{j \in T, i \ne j} \neg Z_j'(x)\big),$$

where $\vec{Z}$ and $\vec{Z'}$ are k-tuples of distinct monadic second-order variables not in τ. We then define $\varphi_P := \mathrm{TC}_{\vec{Z}, \vec{Z'}}[\varphi_T \wedge \varphi_H \wedge \varphi_V](\vec{X}, \vec{Y})$. We claim that $\mathfrak{A}_P \models \varphi_P$ if and only if there exists a corridor tiling for P, from which the claim follows.   ◀

4   Existential positive SO(2TC) collapses to EFO

Let SO(2TC)[∃] denote the syntactic fragment of SO(2TC) in which existential quantifiers and the TC-operator occur only positively, that is, in scope of even number of negations. In this section, we show that the expressive power of SO(2TC)[∃] collapses to that of existential first-order logic ∃FO. In this section, TC-operators are applied only to tuples of second-order variables. As already discussed in the introduction, this restriction is vital: the formula $[\mathrm{TC}_{x,x'}R(x,x') \vee x = x'](y,y')$ expresses reachability in directed graphs, which is not definable even in the full first-order logic.

To facilitate our proofs we start by introducing some helpful terminology.

▶ **Definition 9.** Let $\vec{a}$ and $\vec{b}$ be tuples of the same length and I a set of natural numbers. The *difference* $\mathrm{diff}(\vec{a}, \vec{b})$ of the tuples $\vec{a}$ and $\vec{b}$ is defined as follows

$$\mathrm{diff}(\vec{a}, \vec{b}) := \{i \mid \vec{a}[i] \neq \vec{b}[i]\}.$$

The *similarity* $\mathrm{sim}(\vec{a}, \vec{b})$ of tuples $\vec{a}$ and $\vec{b}$ is defined as follows

$$\mathrm{sim}(\vec{a}, \vec{b}) := \{i \mid \vec{a}[i] = \vec{b}[i]\}.$$

We say that the tuples $\vec{a}$ and $\vec{b}$ are *pairwise compatible* if the sets $\{\vec{a}[i] \mid i \in \mathrm{diff}(\vec{a}, \vec{b})\}$ and $\{\vec{b}[i] \mid i \in \mathrm{diff}(\vec{a}, \vec{b})\}$ are disjoint. The tuples $\vec{a}$ and $\vec{b}$ are *pairwise compatible outside* I if $\{\vec{a}[i] \mid i \in \mathrm{diff}(\vec{a}, \vec{b}), i \notin I\}$ and $\{\vec{b}[i] \mid i \in \mathrm{diff}(\vec{a}, \vec{b}), i \notin I\}$ are disjoint. The tuples $\vec{a}$ and $\vec{b}$ are *pairwise I-compatible* if $\vec{a}$ and $\vec{b}$ are pairwise compatible and $\mathrm{sim}(\vec{a}, \vec{b}) = I$.

▶ **Definition 10.** Let $\sigma \subseteq \tau$ be vocabularies, $\mathfrak{A}$ a τ-structure, and $\vec{a}$ a tuple of elements of A. The (quantifier-free) σ-type of $\vec{a}$ in $\mathfrak{A}$ is the set of those quantifier free $\mathrm{FO}(\sigma)$-formulae $\varphi(\vec{x})$ such that $\mathfrak{A}[\vec{a}/\vec{x}] \models \varphi$.

The following lemma establishes that 2TC-operators that are applied to $\exists\mathrm{FO}$-formulas can be equivalently expressed by the finite $2\mathrm{TC}^m$-operator.

▶ **Lemma 11.** *Every formula φ of the form $[\mathrm{TC}_{\vec{X},\vec{X}'}\theta](\vec{Y}, \vec{Y}')$, where $\theta \in \exists\mathrm{FO}$ and $\vec{X}$, $\vec{X}'$, $\vec{Y}$, $\vec{Y}'$ are tuples of second-order variables, is equivalent with the formula $[\mathrm{TC}^k_{\vec{X},\vec{X}'}\theta](\vec{Y}, \vec{Y}')$, for some $k \in \mathbb{N}$.*

Proof. Let $\theta = \exists x_1 \ldots \exists x_n \psi$, where ψ is quantifier-free, and let τ denote the vocabulary of φ. We will show that for large enough k and for all τ-structures $\mathfrak{A}$

$$\mathfrak{A} \models [\mathrm{TC}_{\vec{X},\vec{X}'}\theta](\vec{Y}, \vec{Y}') \text{ iff } \mathfrak{A} \models [\mathrm{TC}^k_{\vec{X},\vec{X}'}\theta](\vec{Y}, \vec{Y}').$$

From here on we consider τ and φ fixed; especially, by a constant, we mean a number that is independent of the model $\mathfrak{A}$; that is, it may depend on τ and φ.

It suffices to show the left-to-right direction as the converse direction holds trivially for all k. Assume that $\mathfrak{A} \models [\mathrm{TC}_{\vec{X},\vec{X}'}\theta](\vec{Y}, \vec{Y}')$. By the semantics of TC there exists a natural number k_0 and tuples of relations $\vec{B}_0, \ldots, \vec{B}_{k_0}$ on A such that $\vec{B}_0 = \vec{Y}^{\mathfrak{A}}$, $\vec{B}_{k_0} = \vec{Y}'^{\mathfrak{A}}$, and

$$\mathfrak{A}[\vec{B}_i/\vec{X}, \vec{B}_{i+1}/\vec{X}'] \models \theta, \text{ for } 0 \leq i < k_0. \tag{1}$$

It suffices to establish that, if k_0 is large enough, then there exists two natural numbers h and h', $0 \leq h \leq h + 3 \leq h' \leq k_0$, and an interpretation $\vec{H}$ for $\vec{X}$ such that

$$\mathfrak{A}[\vec{B}_h/\vec{X}, \vec{H}/\vec{X}'] \models \theta \text{ and } \mathfrak{A}[\vec{H}/\vec{X}, \vec{B}_{h'}/\vec{X}'] \models \theta.$$

For each $i < k_0$, let $\mathfrak{A}_i := \mathfrak{A}[\vec{B}_i/\vec{X}, \vec{B}_{i+1}/\vec{X}']$ and let σ denote the vocabulary of $\mathfrak{A}_i$. By the semantics of the existential quantifier, (1) is equivalent to saying that

$$\mathfrak{A}_i[\vec{a_i}/x_1, \ldots, x_n] \models \psi, \text{ for } 0 \leq i < k_0, \tag{2}$$

for some n-tuples $\vec{a_0}, \ldots \vec{a_{k_0-1}}$ from A. We will prove the following claim.

Claim. There exists an index set I and $n + 2$ mutually pairwise I-compatible sequences in $\vec{a_1}, \ldots \overrightarrow{a_{k_0-1}}$ that have a common σ-type provided that k_0 is a large enough constant.

Proof of the claim. Let $\vec{c}^0 = (\vec{c_0}^0, \vec{c_1}^0, \ldots, \vec{c_t}^0)$ denote the longest (not necessarily consecutive) subsequence of $\vec{a_1}, \ldots \overrightarrow{a_{k_0-1}}$ that have a common σ-type. Since there are only finitely many σ-types, t can be made as large as needed by making k_0 a large enough constant.

We will next show that there exists $n + 2$ mutually pairwise I-compatible sequences in $\vec{c}^0$ for some I (provided that t is large enough). Set $\mathrm{SIM}_0 := \emptyset$. In the construction below we maintain the following properties for $0 \leq i \leq n$:

- For each $j \in \mathrm{SIM}_i$ and for each tuple $\vec{a}$ and $\vec{b}$ in $\vec{c}^i$ it holds that $\vec{a}[j] = \vec{b}[j]$.
- The length of $\vec{c}^i$ is as long a constant as we want it to be.

For $l < n$, let $\vec{b}_0^l, \ldots, \vec{b}_{t_l}^l$ be a maximal collection (in length) of mutually pairwise SIM_l-compatible sequences from $\vec{c}^l$. If $t_l \geq n+1$ we are done. Otherwise note that, since each $\vec{b}_j^l$ is an n-tuple, the number of different points that may occur in $\vec{b}_0^l, \ldots, \vec{b}_{t_l}^l$ is $\leq n^2 + n$. By an inductive argument we may assume that the length of $\vec{c}^l$ is as large a constant as we want, and thus we may conclude that there exists an index $i \notin \mathrm{SIM}_l$ and an element d_l such that there are as many as we want tuples $\vec{c}_j^l$ in $\vec{c}^l$ such that $\vec{c}_j^l[i] = d_l$. Set $\mathrm{SIM}_{l+1} := \mathrm{SIM}_l \cup \{i\}$ and let $\vec{c}^{l+1}$ be the sequence of exactly those $\vec{a} \in \vec{c}^l$ such that $\vec{a}[i] = d_l$. Notice that the length of $\vec{c}^{l+1}$ is as large a constant as we want it to be.

Finally, the case $l = n$. Note that $\mathrm{SIM}_n = \{0, \ldots, n-1\}$ and $\vec{c}^n$ is a sequence of n-tuples; in fact all tuples in $\vec{c}^n$ are identical. Thus, if the length of $\vec{c}^n$ is at least $n + 2$, the first $n + 2$ sequences of $\vec{c}^n$ constitute a mutually pairwise SIM_n-compatible sequence of length $n + 2$. It is now straightforward but tedious to check how large k_0 has to be so that the length of $\vec{c}^n$ is at least $n + 2$; thus the claim holds. ◄

Now let $\vec{a}_{i_0}, \ldots, \vec{a}_{i_{n+1}}$, $0 < i_0 < \cdots < i_{n+1}$, be mutually pairwise I-compatible sequences from $\vec{a}_1, \ldots \vec{a}_{k_0-1}$ with a common σ-type provided by the Claim. Let $1 \leq j \leq n+1$ be an index such that $\vec{a}_{i_0-1}$ and $\vec{a}_{i_j}$ are pairwise compatible outside I and $\mathrm{sim}(\vec{a}_{i_0-1}, \vec{a}_{i_j}) \subseteq I$. It is straightforward to check that such a j always exists, for if $\vec{a}_{i_0-1}$ and $\vec{a}_{i_{j'}}$ are not pairwise compatible outside I or $\mathrm{sim}(\vec{a}_{i_0-1}, \vec{a}_{i'_j}) \nsubseteq I$, there exists some indices $m, m' \notin I$ such that $\vec{a}_{i_0-1}[m] = \vec{a}_{i_{j'}}[m']$, and for each such $\vec{a}_{i_{j'}}$, the value of the related $\vec{a}_{i_{j'}}[m']$ has to be unique as $\vec{a}_{i_1}, \ldots, \vec{a}_{i_{n+1}}$ are mutually pairwise I-compatible. Now j must exist since the length of $\vec{a}_{i_1}, \ldots, \vec{a}_{i_{n+1}}$ is $n + 1$ while the length of $\vec{a}_{i_0-1}$ is only n.

Consider the models $\mathfrak{A}_{i_0-1} = \mathfrak{A}[\vec{B}_{i_0-1}/\vec{X}, \vec{B}_{i_0}/\vec{X}']$ and $\mathfrak{A}_{i_j} = \mathfrak{A}[\vec{B}_{i_j}/\vec{X}, \vec{B}_{i_{j+i}}/\vec{X}']$ and recall that

$$\mathfrak{A}_{i_0-1}[\vec{a}_{i_0-1}/x_1, \ldots, x_n] \models \psi \text{ and } \mathfrak{A}_{i_j}[\vec{a}_{i_j}/x_1, \ldots, x_n] \models \psi.$$

We claim that there exists a sequence $\vec{B}$ of relations on A such that

$$\mathfrak{A}[\vec{B}_{i_0-1}/\vec{X}, \vec{B}/\vec{X}', \vec{a}_{i_0-1}/x_1, \ldots, x_n] \models \psi \text{ and } \mathfrak{A}[\vec{B}/\vec{X}, \vec{B}_{i_{j+i}}/\vec{X}', \vec{a}_{i_j}/x_1, \ldots, x_n] \models \psi. \quad (3)$$

and thus that $\mathfrak{A}[\vec{B}_{i_0-1}/\vec{X}, \vec{B}/\vec{X}'] \models \theta$ and $\mathfrak{A}[\vec{B}/\vec{X}, \vec{B}_{i_{j+i}}/\vec{X}'] \models \theta$. From this the claim of the theorem follows for $k = k_0$.

It now suffices to show that such a $\vec{B}$ exists. The idea is that $\vec{B}$ looks exactly like $\vec{B}_{i_0}$ with respect to points in $\vec{a}_{i_0-1}$ and like $\vec{B}_{i_j}$ with respect to points $\vec{a}_{i_j}$. Formally $\vec{B}$ is defined as follows. For every relation $\vec{B}[m]$ and tuple $\vec{a} \in A^{\mathrm{ar}(\vec{B}[m])}$

- if $\vec{a}$ is completely included in neither $\vec{a}_{i_0-1}$ nor $\vec{a}_{i_j}$ then we set $\vec{a} \notin \vec{B}[m]$,
- if $\vec{a}$ is completely included in $\vec{a}_{i_0-1}$ then we set $\vec{a} \in \vec{B}[m]$ iff $\vec{a} \in \vec{B}_{i_0}[m]$,
- if $\vec{a}$ is completely included in $\vec{a}_{i_j}$ then we set $\vec{a} \in \vec{B}[m]$ iff $\vec{a} \in \vec{B}_{i_j}[m]$.

Note that if $\vec{a} = (a_1, \ldots, a_m)$ is completely included in both $\vec{a}_{i_0-1}$ and $\vec{a}_{i_j}$ then there exists indices $j_1, \ldots j_m \in I$ such that, for $1 \leq l \leq m$, $a_l = \vec{a}_{i_j}[j_l] = \vec{a}_{i_0}[j_l]$. The former equality follows, with indices in I, since $\vec{a}_{i_0-1}$ and $\vec{a}_{i_j}$ are pairwise compatible outside I and $\operatorname{sim}(\vec{a}_{i_0-1}, \vec{a}_{i_j}) \subseteq I$. The latter equality follows since $\vec{a}_{i_0}$ and $\vec{a}_{i_j}$ are pairwise I-compatible. Since $\vec{a}_{i_0}$ and $\vec{a}_{i_j}$ have the same σ-type $\vec{a} \in \vec{B}_{i_0}[m]$ iff $\vec{a} \in \vec{B}_{i_j}[m]$, for all m, and thus $\vec{B}$ is well-defined. It is now immediate that (3) holds. ◄

▶ **Lemma 12.** *For every formula of vocabulary τ of the form $\exists X \theta$ or $[\mathrm{TC}_{\vec{X},\vec{X}'},\theta](\vec{Y},\vec{Y}')$, where $\theta \in \exists \mathrm{FO}$ and $\vec{X}$, $\vec{X}'$, $\vec{Y}$, $\vec{Y}'$ are tuples of relation variables, there exists an equivalent formula $\varphi \in \exists \mathrm{FO}$ of vocabulary τ.*

Proof. Consider first the formula $\exists X \theta$ (this collapse was remarked, but not proven, by Rosen in the introduction of his paper [35]). Define $n := \operatorname{ar}(X)$ and let k be the number of occurrences of X in θ. The idea behind our translation is that the quantification of X can be equivalently replaced by a quantification of an n-ary relation of size $\leq k$; this can be then expressed in $\exists \mathrm{FO}$ by quantifying k many n-tuples (content of the finite relation).

Let $\theta_\emptyset$ denote the formula obtained from θ by replacing every occurrence of the relation variable X of the form $X(\vec{x})$ in θ by the formula $\exists x(x \neq x)$. Define

$$\gamma := \exists \vec{x}_1 \ldots \exists \vec{x}_k (\theta_\emptyset \vee \theta'),$$

where, for each i, $\exists \vec{x}_i$ is a shorthand for $\exists x_{1,i} \ldots \exists x_{n,i}$ and θ' is the formula obtained from θ by substituting each occurrence of the relation variable X of the form $X(\vec{x})$ in θ by $\bigvee_{1 \leq i \leq k}(\vec{x} = \vec{x}_i)$. It is straightforward to check that γ is an $\exists \mathrm{FO}$-formula of vocabulary τ equivalent with $\exists X \theta$.

Consider then the formula $\varphi = [\mathrm{TC}_{\vec{X},\vec{X}'},\theta](\vec{Y},\vec{Y}')$. In order to simplify the presentation, we stipulate that $\vec{X}$ and $\vec{X}'$ are of length one, that is, variables X and X', respectively; the generalisation of the proof for arbitrary tuples of second-order variables is straightforward. By Lemma 11, we obtain $k \in \mathbb{N}$ such that φ and $\varphi' := [\mathrm{TC}_{X,X'}^k \theta](Y,Y')$ are equivalent.

The following formulas are defined via substitution; by $\theta(A/B)$ we denote the formula obtained from θ by substituting each occurrence of the symbol B by the symbol A.

- $\theta_0^{\mathrm{end}} := \theta(Y/X, Y'/X')$ and $\theta_i^{\mathrm{end}} := \theta(X_i/X, Y'/X')$, for $1 \leq i < k$,
- $\theta_1^{\mathrm{move}} := \theta(Y/X, X_1/X')$ and $\theta_i^{\mathrm{move}} := \theta(X_{i-1}/X, X_i/X')$, for $2 \leq i < k$.

Let ψ denote the following formula of existential second-order logic

$$\exists X_1 \ldots \exists X_{k-1} \bigvee_{0 \leq n < k} (\theta_n^{\mathrm{end}} \wedge \bigwedge_{1 \leq i \leq n} \theta_i^{\mathrm{move}}).$$

It is immediate that φ' and ψ are equivalent. Note that ψ is of the form $\exists X_1 \ldots \exists X_{k-1} \psi'$, where ψ' is an $\exists \mathrm{FO}$-formula. By repetitively applying the first case of this lemma to subformulas of ψ, we eventually obtain an equivalent $\exists \mathrm{FO}$-formula over τ as required. ◄

The following theorem now follows by applying Lemma 12 repetitively bottom up.

▶ **Theorem 13.** *The expressive powers of $\mathrm{SO}(2\mathrm{TC})[\exists]$ and $\exists \mathrm{FO}$ coincide.*

5 MSO(TC) and counting

We define a counting extension of MSO(TC) and show that the extension does not add expressive power to the logic. In this way, we demonstrate that quite a bit of queries involving counting can be expressed already in MSO(TC).

5.1 Syntax and semantics of CMSO(TC)

We assume a sufficient supply of *counter variables* or simply *counters*, which are a new sort of variables. We use the Greek letters μ and ν (with subscripts) to denote counter variables. The notion of a vocabulary is extended so that it may also contain counters. A structure $\mathfrak{A}$ over a vocabulary τ is defined to be a pair (A, I) as before, where I now also maps the counters in τ to elements of $\{0, \ldots, n\}$, where n is the cardinality of A.

We also assume a sufficient supply of *numeric predicates*. Intuitively numeric predicates are relations over natural numbers such as the tables of multiplication and addition. Technically, we use an approach similar to generalised quantifiers; a k-ary numeric predicate is a class $Q_p \subseteq \mathbb{N}^{k+1}$ of $k + 1$-tuples of natural numbers. For a numeric predicate Q_p, we use p as a symbol referring to the predicate. For simplicity, we often call p also numeric predicate. Note that when evaluating a k-ary numeric predicate $p(\mu_1, \ldots, \mu_k)$ on a finite structure $\mathfrak{A}$, we let the numeric predicate Q_p access also the cardinality of the structure in question, and thus Q_p consists of $k + 1$-tuples and not k-tuples. This convention allows us, for example, to regard the modular sum $a + b \equiv c \pmod n$, where n refers to the cardinality of the structure, as a 3-ary numeric predicate.

We consider only those numeric predicates which can be decided in NLOGSPACE. Since, on finite ordered structures, first-order transitive closure logic captures NLOGSPACE, this boils down to being definable in first-order transitive closure logic when the counter variables are interpreted as points in an ordered structure representing an initial segment of natural numbers (see Definition 16 and Proposition 17 below for precise formulations). Note that the equality of numeric variables is also a 2-ary NLOGSPACE predicate.

▶ **Definition 14.** The syntax of CMSO(TC) extends the syntax of MSO(TC) as follows:

- Let φ be a formula, μ a counter, and x a first-order variable. Then $\mu = \#\{x \mid \varphi\}$ is also a formula. The set of its free variables is defined to be $(\mathrm{FV}(\varphi) - \{x\}) \cup \{\mu\}$.
- If φ is a formula and μ a counter then also $\exists \mu\, \varphi$ is a formula with set of free variables $\mathrm{FV}(\varphi) - \{\mu\}$.
- Let $\mu_1, \ldots, \mu_k$ be counters and let p be a k-ary numeric predicate. Then $p(\mu_1, \ldots, \mu_k)$ is a formula with the set of free variables $\{\mu_1, \ldots, \mu_k\}$.
- The scope of the transitive-closure operator is widened to apply as well to counters. Formally, in a formula of the form $[\mathrm{TC}_{\vec{X}, \vec{X}'}\varphi](\vec{Y}, \vec{Y}')$, the variables in $\vec{X}$, $\vec{X}'$, $\vec{Y}$, and $\vec{Y}'$ may also include counters. We still require that the tuples $\vec{X}$, $\vec{X}'$, $\vec{Y}$, and $\vec{Y}'$ have the same sort, i.e., if a counter appears in some position in one of these tuples then a counter must appear in that position in each of the tuples.

▶ **Definition 15.** The satisfaction relation, $\mathfrak{A} \models \psi$, for CMSO(TC) formulas ψ and structures $\mathfrak{A} = (A, I)$ over a vocabulary appropriate for ψ is defined in the same way as for MSO(TC) with the following additional clauses.

- Let ψ be of the form $\exists \mu\, \varphi$, where μ is a counter, and let n denote the cardinality of A. Then $\mathfrak{A} \models \psi$ iff there exists a number $i \in \{0, \ldots, n\}$ such that $\mathfrak{A}[i/\mu] \models \varphi$.
- Let ψ be of the form $\mu = \#\{x \mid \varphi\}$. Then $\mathfrak{A} \models \psi$ iff $I(\mu)$ equals the cardinality of the set $\{a \in A \mid \mathfrak{A}[a/x] \models \varphi\}$.
- Let ψ be of the form $p(\mu_1, \ldots, \mu_k)$, where $\mu_1, \ldots, \mu_k$ are counters and p is a k-ary numeric predicate. Then $\mathfrak{A} \models p(\mu_1, \ldots, \mu_k)$ iff $\big(|A|, I(\mu_1), \ldots, I(\mu_k)\big) \in Q_p$.

▶ **Definition 16.** A k-ary numeric predicate Q_p is *decidable in* NLOGSPACE if the membership $(n_0, \ldots, n_k) \in Q_p$ can be decided by a nondeterministic Turing machine that uses logarithmic space when the numbers $n_0, \ldots, n_k$ are given in unary. Note that this is equivalent to linear space when $n_0, \ldots, n_k$ are given in binary.

From now on we restrict our attention to numeric predicates that are decidable in NLOGSPACE. The following proposition follows directly from a result of Immerman (Theorem 5) that, on ordered structures, FO(1TC) captures NLOGSPACE.

▶ **Proposition 17.** *For every k-ary numeric predicate Q_p decidable in NLOGSPACE there exists a formula φ_p of FO(1TC) over $\{s, x_1, \ldots, x_k\}$, where s is a binary second-order variable and $x_1, \ldots, x_k$ are first-order variables, s.t. for all appropriate structures $\mathfrak{A}$ for $p(\mu_1, \ldots, \mu_k)$*

$$\mathfrak{A} \models p(\mu_1, \ldots, \mu_k) \text{ iff } \big(|A|, I(\mu_1), \ldots, I(\mu_k)\big) \in Q_p \text{ iff } (B, J) \models \varphi_p,$$

where $B = \{0, 1, \ldots, |A|\}$, $J(s)$ is the successor relation of B, and $J(x_i) = I(\mu_i)$, for $1 \le i \le k$.

5.2 CMSO(TC) collapses to MSO(TC)

Let τ be a vocabulary with counters. Let τ^* denote the vocabulary without counters obtained from τ by viewing each counter variable of τ as a set variable. Let $\mathfrak{A} = (A, I)$ be a structure over τ, and let $\mathfrak{B} = (A, J)$ be a structure over τ^* with the same domain as $\mathfrak{A}$. We say that $\mathfrak{B}$ *simulates* $\mathfrak{A}$ if for every counter μ in τ, the set $J(\mu)$ has cardinality $I(\mu)$, and $J(X) = I(X)$, for each first-order or second-order variable $X \in \tau$. Let φ be a CMSO(TC)-formula over τ and ψ an MSO(TC) formula over τ^*. We say that ψ *simulates* φ if whenever $\mathfrak{B}$ simulates $\mathfrak{A}$, we have that $\mathfrak{A} \models \varphi$ if and only if $\mathfrak{B} \models \psi$.

Let $\varphi(x)$ and $\psi(y)$ be formulae of some logic. The *Härtig quantifier* is defined as follows:

$$\mathfrak{A} \models \mathrm{H}xy(\varphi(x), \psi(y)) \Leftrightarrow \text{the sets } \{a \in A \mid \mathfrak{A}[a/x] \models \varphi\} \text{ and } \{b \in A \mid \mathfrak{A}[b/y] \models \psi\}$$
$$\text{have the same cardinality}$$

▶ **Proposition 18.** *The Härtig quantifier can be expressed in MSO(TC).*

Proof. Consider a structure (A, I) and monadic second-order variables X, Y, X' and Y'. Let $\psi_{\mathrm{decrement}}$ denote an FO-formula expressing that $I(X') = I(X) \setminus \{a\}$ and $I(Y') = I(Y) \setminus \{b\}$, for some $a \in I(X)$ and $b \in I(Y)$. Define

$$\psi_{\mathrm{ec}} := \exists X_\emptyset \Big(\big(\forall x \neg X_\emptyset(x)\big) \wedge [\mathrm{TC}_{X,Y,X',Y'} \psi_{\mathrm{decrement}}](Z, Z', X_\emptyset, X_\emptyset) \Big).$$

It is straightforward to check that ψ_{ec} holds in (A, I) if and only if $|I(Z)| = |I(Z')|$. Therefore $\mathrm{H}xy(\varphi(x), \psi(y))$ is equivalent with the formula

$$\exists Z \exists Z' \big(\forall x(\varphi(x) \leftrightarrow Z(x)) \wedge \forall y(\psi(y) \leftrightarrow Z'(y)) \wedge \psi_{\mathrm{ec}} \big),$$

assuming that Z, Z' are variable symbols that occur in neither φ nor ψ. ◀

▶ **Lemma 19.** *Let $\tau = \{s, x_1, \ldots, x_n\}$ and $\sigma = \{X_1, \ldots, X_n\}$ be vocabularies, where s is a binary second-order variable, $x_1, \ldots, x_n$ are first-order variables, and $X_1, \ldots, X_n$ are monadic second-order variables. For every FO(1TC)-formula φ over τ there exists an MSO(TC)-formula φ^+ over σ such that*

$$(A, I) \models \varphi \quad \Leftrightarrow \quad (B, J) \models \varphi^+,$$

for every (A, I) and (B, J) such that (A, I) is a structure over vocabulary τ, where $A = \{0, \ldots, m\}$, for some $m \in \mathbb{N}$, and $I(s)$ is the canonical successor relation on A, and (B, J) is a structure over vocabulary σ such that $|B| = m$ and $|J(X_i)| = I(x_i)$, for $1 \le i \le n$.

Proof. We define the translation $^+$ recursively as follows. In the translation, we introduce for each first-order variable x_i a monadic second-order variable X_i by using the corresponding capital letter with the same index. Consequently, in tuples of variables, identities between the variables are maintained. The idea of the translation is that natural numbers i are simulated by sets of cardinality i. Identities between first-order variables are then simulated with the help of the Härtig quantifier, which, by Proposition 18, is definable in MSO(TC).

- For ψ of the form $x_i = x_j$, define $\psi^+ := \mathrm{H}xy\big(X_i(x), X_j(y)\big)$.
- For ψ of the form $s(x_i, x_j)$, define $\psi^+ := \exists z\Big(\neg X_i(z) \wedge \mathrm{H}xy\big(X_i(x) \vee x = z, X_j(y)\big)\Big)$.
- For ψ of the form $\neg\varphi$ and $(\varphi \wedge \theta)$, define ψ^+ as $\neg\varphi^+$ and $(\varphi^+ \wedge \theta^+)$, respectively.
- For ψ of the form $\exists x_i \varphi$, define $\psi^+ := \exists X_i \varphi^+$, where X_i is a monadic second-order variable.
- For ψ of the form $[\mathrm{TC}_{\vec{x},\vec{x'}}\varphi](\vec{y}, \vec{y'})$, define $\psi^+ := [\mathrm{TC}_{\vec{X},\vec{X'}}\varphi^+](\vec{Y}, \vec{Y'})$, where $\vec{X}$, $\vec{X'}$, $\vec{Y}$, and $\vec{Y'}$ are tuples of monadic second-order variables that correspond to the tuples $\vec{x}$, $\vec{x'}$, $\vec{y}$, and $\vec{y'}$ of first-order variables.

The correctness of the translation follows by a simple inductive argument. ◄

With the help of the previous lemma, we are now ready to show how CMSO(TC)-formulas can be simulated in MSO(TC).

▶ **Theorem 20.** *Every* CMSO(TC)*-formula can be simulated by an* MSO(TC)*-formula.*

Proof. Let τ be a vocabulary with counters and τ^* the vocabulary without counters obtained from τ by viewing each counter as a set variable. We define recursively a translation * that maps CMSO(TC)-formulas over vocabulary τ to MSO(TC)-formulas over τ^*.

- For ψ of the form $x_i = x_j$, define $\psi^* := x_i = x_j$.
- For ψ of the form $X(x_1, \ldots, x_n)$, define $\psi^* := X(x_1, \ldots, x_n)$.
- For an NLOGSPACE numeric predicate Q_p and ψ be of the form $p(\mu_1, \ldots, \mu_k)$, define ψ^* as $\varphi_p^+(\mu_1/X_1, \ldots, \mu_k/X_k)$, where $^+$ is the translation defined in Lemma 19 and φ_p the defining formula of Q_p obtained from Proposition 17.
- For ψ of the form $\mu = \#\{x \mid \varphi\}$, define ψ^* as the MSO(TC)-formula $\mathrm{H}xy(\varphi^*, \mu(y))$.
- For ψ of the form $\neg\varphi$ and $(\varphi \wedge \theta)$, define ψ^* as $\neg\varphi^*$ and $(\varphi^* \wedge \theta^*)$, respectively.
- For ψ of the form $\exists x_i \varphi$, $\exists \mu_i \varphi$, and $\exists X_i \varphi$, define ψ^* as $\exists x_i \varphi^*$, $\exists \mu_i \varphi^*$, and $\exists X_i \varphi^*$. Remember that, on the right, μ_i is treated a as a monadic second-order variable.
- For ψ of the form $[\mathrm{TC}_{\vec{X},\vec{X'}}\varphi](\vec{Y}, \vec{Y'})$, define $\psi^* := [\mathrm{TC}_{\vec{X},\vec{X'}}\varphi^*](\vec{Y}, \vec{Y'})$.

We claim that, for every CMSO(TC)-formula ψ over τ, ψ^* is an MSO(TC)-formula over τ^* that simulates ψ. Correctness of the simulation follows by induction using Lemma 19 and Proposition 17.

We show the case for the numeric predicates. Let $\mathfrak{A} = (A, I)$ be a τ-structure and $\mathfrak{A}^*$ a τ^*-structure that simulates $\mathfrak{A}$. Let Q_p be a k-ary NLOGSPACE numeric predicate, $\mu_1, \ldots, \mu_k$ counters from τ, and φ_p the defining FO(1TC)-formula of Q_p given by Proposition 17. Then, by Proposition 17,

$$\mathfrak{A} \models p(\mu_1, \ldots, \mu_k) \text{ iff } (B, J) \models \varphi_p,$$

where $B = \{0, 1, \ldots, |A|\}$, $J(s)$ is the successor relation of B, and $J(x_i) = I(\mu_i)$, for $1 \leq i \leq k$. Let $^+$ denote the translation from FO(1TC) to MSO(TC) defined in Lemma 19. Then, by Lemma 19, it follows that $(B, J) \models \varphi_p$ iff $\mathfrak{A} \models \varphi_p^+$. ◄

In the next example, we introduce notation for some MSO(TC)-definable numeric predicates that are used in the following sections.

▶ **Example 21.** Let k be a natural number, $X, Y, Z, X_1, \ldots, X_n$ monadic second-order variables, and $\mathfrak{A} = (A, I)$ an appropriate structure. The following numeric predicates are clearly NLOGSPACE-definable and thus, by Theorem 20, definable in MSO(TC):

- $\mathfrak{A} \models \mathrm{size}(X, k)$ iff $|I(X)| = k$,
- $\mathfrak{A} \models \times(X, Y, Z)$ iff $|I(X)| \times |I(Y)| = |I(Z)|$,
- $\mathfrak{A} \models +(X_1, \ldots, X_n, Y)$ iff $|I(X_1)| + \cdots + |I(X_n)| = |I(Y)|$.

6 Order-invariant MSO

Order-invariance plays an important role in finite model theory. In descriptive complexity theory many characterisation rely on the existence of a linear order. However the particular order in a given stricture is often not important. Related to applications in computer science, it is often possible to access an ordering of the structure that is not controllable and thus a use of the ordering should be such that change in the ordering should not make a difference. Consequently, in both cases order can be used, but in a way that the described properties are *order-invariant*.

Let $\tau_\leq := \tau \cup \{\leq\}$ be a finite vocabulary, where $\leq$ is a binary relation symbol. A formula $\varphi \in$ MSO over $\tau_\leq$ is *order-invariant*, if for every τ-structure $\mathfrak{A}$ and expansions $\mathfrak{A}'$ and $\mathfrak{A}^*$ of $\mathfrak{A}$ to the vocabulary $\tau_\leq$, in which $\leq^{\mathfrak{A}'}$ and $\leq^{\mathfrak{A}^*}$ are linear orders of A, we have that $\mathfrak{A}' \models \varphi$ if and only if $\mathfrak{A}^* \models \varphi$. A class $\mathcal{C}$ of τ-structures is *definable in order-invariant* MSO if and only if the class $\{(\mathfrak{A}, \leq) \mid \mathfrak{A} \in \mathcal{C} \text{ and } \leq \text{ is a linear order of } A\}$ is definable by some order-invariant MSO-formula.

We call a vocabulary τ *a unary vocabulary* if it consists of only monadic second-order variables. In this section we establish that on unary vocabularies MSO(TC) is strictly more expressive than order-invariant MSO. The separation holds already for the empty vocabulary.

6.1 Separation on empty vocabulary

First note that over vocabulary $\{\leq\}$ there exists only one structure, up to isomorphism, of size k, for each $k \in N$, in which $\leq$ is interpreted as a linear order of the domain. Consequently, every MSO-formula of vocabulary $\{\leq\}$ is order-invariant. Also note that, in fact, $\{\leq\}$-structures interpreted as word models correspond to finite strings over some fixed unary alphabet. Thus, via Büchi's theorem, we obtain that, over the empty vocabulary, order-invariant MSO captures essentially regular languages over unary alphabets. Hence, to separate MSO(TC) from order-invariant MSO over the empty vocabulary, it suffices to observe that not all NLOGSPACE properties of unary strings are regular (recall Theorem 5 and Lemma 19). The following example gives a concrete example of the separation.

▶ **Example 22.** Consider the class $\mathcal{C} = \{\mathfrak{A} \mid |A| \text{ is a prime number}\}$ of $\emptyset$-structures. Clearly the language of prime length words over some unary alphabet is not regular and thus it follows via Büchi's theorem that $\mathcal{C}$ is not definable in order-invariant MSO. However the following formula of MSO(TC) defines $\mathcal{C}$. We use MSO(TC)-definable numeric predicates introduced in Example 21.

$$\exists X \forall Y \forall Z \big(\forall x (X(x)) \wedge (\mathrm{size}(Y, 1) \vee \mathrm{size}(Z, 1) \vee \neg \times (Y, Z, X))\big) \wedge \exists x \exists y \, \neg x = y.$$

▶ **Corollary 23.** *For any vocabulary τ, there exists a class $\mathcal{C}$ of τ-structures such that $\mathcal{C}$ is definable in* MSO(TC) *but it is not definable in order-invariant* MSO.

6.2   Inclusion on unary vocabularies

We will show that every class of structures over a unary vocabulary τ that is definable in order-invariant MSO is also definable in MSO(TC).

▶ **Definition 24.** For a finite word w of some finite alphabet $\Sigma = \{a_1 \ldots, a_k\}$, a *Parikh vector $p(w)$ of w* is the k-tuple $(|w|_{a_1}, \ldots, |w|_{a_k})$ where $|w|_{a_i}$ denotes the number of a_is in w. A *Parikh image $P(L)$ of a language L* is the set $\{p(w) \mid w \in L\}$ of Parikh vectors of the words in the language.

A subset S of $\mathbb{N}^k$ is a *linear set* if $S = \{\vec{v}_0 + \sum_{i=1}^{m} a_i \vec{v}_i \mid a_1, \ldots, a_m \in \mathbb{N}\}$ for some *offset* $\vec{v}_0 \in \mathbb{N}^k$ and *generators* $\vec{v}_1, \ldots, \vec{v}_m \in \mathbb{N}^k$.

▶ **Theorem 25** (Parikh's theorem, [30])**.** *For every regular language L its Parikh image $P(L)$ is a finite union of linear sets.*

We use the following improved version of Parikh's theorem:

▶ **Theorem 26** ([26])**.** *For every regular language L over alphabet of size k its Parikh image $P(L)$ is a finite union of linear sets with at most k generators.*

▶ **Definition 27.** Let $\tau = \{X_1, \ldots, X_k\}$ be a finite unary vocabulary and let $Y_1, \ldots, Y_{2^k}$ denote the Boolean combinations of the variables in τ in some fixed order. For every structure $\mathfrak{A} = (A, I)$ over τ, we extend the scope of I to include also $Y_1, \ldots, Y_{2^k}$ in the obvious manner. The *Parikh vector $p(\mathfrak{A})$ of $\mathfrak{A}$* is the 2^k-tuple $\big(|I(Y_1)|, \ldots, |I(Y_{2^k})|\big)$. A *Parikh image $P(\mathcal{C})$ of a class of τ-structures $\mathcal{C}$* is the set $\{p(\mathfrak{A}) \mid \mathfrak{A} \in \mathcal{C}\}$.

▶ **Theorem 28.** *Over finite unary vocabularies* MSO(TC) *is strictly more expressive than order-invariant* MSO*.*

Proof. Strictness follows directly from Corollary 23 and thus it suffices to establish inclusion. Let $\tau = \{X_1, \ldots, X_k\}$ be a finite unary vocabulary and φ an order-invariant MSO-formula of vocabulary $\tau_\leq$. Let $\mathcal{C}$ be the class of τ structures that φ defines. We will show that $\mathcal{C}$ is definable in MSO(TC). Set $n := 2^k$ and let $Y_1, \ldots, Y_n$ denote the Boolean combinations of the variables in τ in some fixed order; we regard these combinations also as fresh monadic second-order variables and set $\sigma := \{Y_1, \ldots, Y_n\}$. For each X_i, let χ_i denote the disjunction of those variables Y_j in which X_i occurs positively. Let $\mathcal{C}_\leq$ denote the class of $\tau_\leq$-structures that φ defines. We may view $\mathcal{C}_\leq$ also as a language L over the alphabet σ and as the class L_w of $\sigma_\leq$-structures corresponding to the word models of the language L. Let φ^* denote the order-invariant MSO-formula over $\sigma_\leq$ obtained from φ by substituting each variable X_i by the formula χ_i. Since φ^* clearly defines L_w, by Büchi's Theorem, L is regular. Consequently, by the improved version of Parikh's Theorem (Theorem 26), the Parikh image P(L) of L is a finite union of linear sets with at most n generators.

Observe that if two τ-structures have the same Parikh image, the structures are isomorphic. Thus $\mathcal{C}$ is invariant under Parikh images. Hence $\mathcal{C}$ is uniquely characterised by its Parikh image $P(\mathcal{C})$, which, since $P(L) = P(\mathcal{C})$, is a finite union of linear sets with at most n generators.

Claim.   For every linear set $A \subseteq \mathbb{N}^n$, where $n = 2^k$, there exists a formula φ_A of MSO(TC) of vocabulary $\tau = \{X_1, \ldots X_k\}$ such that φ_A defines the class of τ-structures that have A as their Parikh image.

With the help of the above claim, the theorem follows in a straightforward manner. Let $A_1, \ldots, A_m$ be a finite collection of linear sets such that $P(\mathcal{C}) = A_1 \cup \cdots \cup A_m$ and let $\varphi_{A_1}, \ldots, \varphi_{A_m}$ be the related MSO(TC)-formulas of vocabulary τ provided by the claim. Clearly $\psi := \varphi_{A_1} \vee \cdots \vee \varphi_{A_m}$ defines $\mathcal{C}$.

Proof of the Claim. Let $A \subseteq \mathbb{N}^n$ be a linear set with n generators, i.e.,

$$A = \{\vec{v}_0 + \sum_{j=1}^{n} a_j \vec{v}_j \mid a_1, \ldots, a_n \in \mathbb{N}\}, \text{ for some } \vec{v}_0, \vec{v}_1, \ldots, \vec{v}_n \in \mathbb{N}^n.$$

For each tuple $\vec{v} \in \mathbb{N}^n$ and n-tuple of monadic second-order variables $\vec{Z}$, let $\text{size}(\vec{Z}, \vec{v})$ denote the FO-formula stating that, for each i, the size of the extension of $\vec{Z}[i]$ is $\vec{v}[i]$. For $0 \leq i \leq n$, we introduce fresh distinct n-tuples of monadic variable symbols $\vec{Z}_i$ and define

$$\varphi_{\text{gen}} := \bigwedge_{0 \leq i \leq n} \text{size}(\vec{Z}_i, \vec{v}_i).$$

Let $\vec{R}_1, \ldots \vec{R}_n$ be fresh distinct n-tuples of monadic second-order variables and let $S_1, \ldots, S_n$ be fresh distinct monadic second-order variables. Define

$$\varphi_A^* := \exists \vec{Z}_0 \ldots \vec{Z}_n \vec{R}_1 \ldots \vec{R}_n S_1 \ldots S_n \, \varphi_{\text{gen}} \wedge$$
$$\bigwedge_{1 \leq i,j \leq n} \times (\vec{Z}_i[j], S_i, \vec{R}_i[j]) \wedge \bigwedge_{1 \leq i \leq n} +(\vec{Z}_0[i], \vec{R}_1[i], \ldots, \vec{R}_n[i], Y_i), \quad (4)$$

where $\times$ and $+$ refer to the MSO(TC)-formulas defined in Example 21. Finally define $\varphi_A := \exists Y_1 \ldots Y_n \varphi_{BC} \wedge \varphi_A^*$, where φ_{BC} is an FO-formula stating that, for each i, the extension of Y_i is the extension of the Boolean combination of the variables in τ that Y_i represents. A τ-structure $\mathfrak{B}$ satisfies φ_A if and only if the Parikh image of $\mathfrak{B}$ is A. ◀

◀

7 Conclusion

There are quite a number of interesting challenging questions regarding the expressive power within second-order transitive-closure logic.

1. We have shown that MSO(TC) can do counting, and thus can certainly express some queries not expressible in fixpoint logic FO(LFP). A natural question is whether MSO(TC) can also be separated from the counting extension of FO(LFP). Note that MSO(TC) can express numerical predicates in NLOGSPACE, while counting fixpoint logic can express numerical predicates in PTIME. Thus, over the empty vocabulary, the question seems related to a famous open problem from complexity theory. Note however, that it is not even clear that MSO(TC) can *only* express numerical predicates in NLOGSPACE. Over graphs, the answer is probably affirmative as the CFI query can probably be expressed in MSO(TC).

2. The converse question, whether there is a fixpoint logic query not expressible in MSO(TC), is fascinating. On ordered structures, this would show that there are problems in PTIME that are not in NLIN, which is open (we only know that the two classes are different [29]). On unordered structures, however, we actually conjecture that the query about a binary relation (transition system) R and two nodes a and b, that asks whether a and b are *bisimilar* w.r.t. R, is not expressible in MSO(TC).

3. In stating Theorem 5 we recalled that SO(arity k)(TC) captures the complexity class NSPACE(n^k), on strings. What about ordered structures in general? Using the standard adjacency matrix encoding of a relational structure as a string [25], it follows that on ordered structures over vocabularies with maximal arity a, SO(arity $k \cdot a$)(TC) can express

all queries in $\text{NSPACE}(n^k)$. Can we show that this blowup in arity is necessary? For example, can we show that MSO(TC) does *not* capture NLIN over ordered graphs (binary relations)?

4. In the previous section we have clarified the relationship between MSO(TC) and order-invariant MSO, over unary vocabularies. What about higher arities?

References

1 Serge Abiteboul and Victor Vianu. Fixpoint extensions of first-order logic and datalog-like languages. In *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, pages 71–79. IEEE Computer Society, 1989. `doi:10.1109/LICS.1989.39160`.

2 Jean-Raymond Abrial. *The B-book - Assigning programs to meanings*. Cambridge University Press, 2005.

3 Jean-Raymond Abrial. *Modeling in Event-B - System and Software Engineering*. Cambridge University Press, 2010.

4 Faisal N. Abu-Khzam and Michael A. Langston. Graph coloring and the immersion order. In *Computing and Combinatorics, 9th Annual International Conference (COCOON 2003)*, pages 394–403, 2003.

5 Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive Datalog, CSP, and MMSNP. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '13, pages 213–224, New York, NY, USA, 2013. ACM. `doi:10.1145/2463664.2465223`.

6 Andreas Blass, Yuri Gurevich, and Jan Van den Bussche. Abstract state machines and computationally complete query languages. *Information and Computation*, 174(1):20–36, 2002. `doi:10.1006/inco.2001.3067`.

7 Andreas Blass, Yuri Gurevich, and Saharon Shelah. Choiceless polynomial time. *Annals of Pure and Applied Logic*, 100(1):141–187, 1999. `doi:10.1016/S0168-0072(99)00005-6`.

8 Joshua Blinkhorn and Olaf Beyersdorff. Shortening QBF proofs with dependency schemes. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 263–280, Cham, 2017. Springer International Publishing.

9 Béla Bollobás. *Modern Graph Theory*, volume 184 of *Graduate Texts in Mathematics*. Springer, 2002.

10 E. Börger and R. F. Stärk. *Abstract State Machines. A Method for High-Level System Design and Analysis*. Springer, 2003.

11 Pierre Bourhis, Markus Krötzsch, and Sebastian Rudolph. Reasonable highly expressive query languages - IJCAI-15 distinguished paper (honorary mention). In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2826–2832. AAAI Press, 2015.

12 Bogdan S Chlebus. Domino-tiling games. *J. Comput. Syst. Sci.*, 32(3):374–392, 1986. `doi:10.1016/0022-0000(86)90036-X`.

13 Bruno Courcelle. The monadic second-order logic of graphs VIII: orientations. *Ann. Pure Appl. Logic*, 72(2):103–143, 1995. `doi:10.1016/0168-0072(95)94698-V`.

14 Stephen Dill, Ravi Kumar, Kevin S. Mccurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the web. *ACM Trans. Internet Technol.*, 2(3):205–223, 2002.

15 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory*. Perspectives in Mathematical Logic. Springer, 1995.

16   Flavio Ferrarotti. *Expressibility of Higher-Order Logics on Relational Databases: Proper Hierarchies.* PhD thesis, Massey University, Wellington, New Zealand, 2008. URL: `http://hdl.handle.net/10179/799`.

17   Flavio Ferrarotti, Senén González, and José Maria Turull Torres. On fragments of higher order logics that on finite structures collapse to second order. In Juliette Kennedy and Ruy J. G. B. de Queiroz, editors, *Logic, Language, Information, and Computation - 24th International Workshop, WoLLIC 2017, London, UK, July 18-21, 2017, Proceedings*, volume 10388 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2017. `doi:10.1007/978-3-662-55386-2_9`.

18   Flavio Ferrarotti, Wei Ren, and Jose Maria Turull Torres. Expressing properties in second- and third-order logic: hypercube graphs and SATQBF. *Logic Journal of the IGPL*, 22(2):355–386, 2014. `doi:10.1093/jigpal/jzt025`.

19   Martin Grohe, Kenichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 479–488. ACM, 2011.

20   R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Phys. Rev. E*, 68:065103, Dec 2003.

21   David Harel and David Peleg. On static logics, dynamic logics, and complexity classes. *Information and Control*, 60(1-3):86–102, 1984. `doi:10.1016/S0019-9958(84)80023-6`.

22   Marijn J. H. Heule, Martina Seidl, and Armin Biere. Solution validation and extraction for QBF preprocessing. *J. Autom. Reasoning*, 58(1):97–125, 2017. `doi:10.1007/s10817-016-9390-4`.

23   Neil Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, aug 1987. `doi:10.1137/0216051`.

24   Neil Immerman. Nondeterministic space is closed under complementation. *SIAM J. Comput.*, 17(5):935–938, 1988. `doi:10.1137/0217058`.

25   Neil Immerman. *Descriptive Complexity.* Springer, 1998.

26   E. Kopczynski and A. W. To. Parikh images of grammars: Complexity and applications. In *2010 25th Annual IEEE Symposium on Logic in Computer Science*, pages 80–89, July 2010. `doi:10.1109/LICS.2010.21`.

27   Richard Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput.*, 6:467–480, 1977.

28   Leslie Lamport. *Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers.* Addison-Wesley, 2002.

29   C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

30   Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, 1966. `doi:10.1145/321356.321364`.

31   Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Dependency learning for QBF. In Serge Gaspers and Toby Walsh, editors, *Theory and Applications of Satisfiability Testing – SAT 2017*, pages 298–313, Cham, 2017. Springer International Publishing.

32   Albert Réka. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.

33   Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. Regular queries on graph databases. In *18th International Conference on Database Theory, ICDT 2015, March 23-27, 2015, Brussels, Belgium*, pages 177–194, 2015. `doi:10.4230/LIPIcs.ICDT.2015.177`.

34   David Richerby. Logical characterizations of PSPACE. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, 18th International Workshop, CSL 2004, 13th Annual Conference of the EACSL, Karpacz, Poland, September 20-24, 2004, Proceedings*, volume 3210 of *Lecture Notes in Computer Science*, pages 370–384. Springer, 2004. `doi:10.1007/978-3-540-30124-0_29`.

35   E. Rosen. An existential fragment of second order logic. *Archive for Mathematical Logic*, 38(4–5):217–234, 1999.

36   Sebastian Rudolph and Markus Krötzsch. Flag & check: Data access with monadically defined queries. In *Proceedings of the 32Nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '13, pages 151–162, New York, NY, USA, 2013. ACM. `doi:10.1145/2463664.2465227`.

37   Chaoming Song, Shlomo Havlin, and Hernán A. Makse. Self-similarity of complex networks. *Nature*, 433:392–395, 2005.