Lower Bounds for Meta-Reconfiguration

Kord Eickmeyer
Technische Universität Darmstadt, Germany

Tatsuya Gima
Hokkaido University, Sapporo, Japan

Michael Lampis
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Valia Mitsou
Université Paris Cité, IRIF, CNRS, 75205, Paris, France

Edouard Nemery
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Yota Otachi
Nagoya University, Japan

Manolis Vasilakis
Université Paris-Dauphine, PSL University, CNRS UMR7243, LAMSADE, Paris, France

Daniel Vaz
LIGM, Université Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France

Abstract

In this paper, we explore the limits of algorithmic meta-theorems for combinatorial reconfiguration on graphs and prove several intractability results for highly restricted cases, which tightly complement the positive results by Mouawad et al. [IPEC 2014] and Gima et al. [Algorithmica 2024]. In this setting, we study reconfiguration problems on graphs in which the feasible sets are defined by formulas of first-order or monadic second-order logic: for a formula $\varphi(X)$ with a free set variable X, the problem asks whether two given sets are connected by a token-jumping sequence in which every set satisfies φ on the input graph.

Our main contribution is to show that the problem is intractable even for first-order logic and for severely restricted graphs, such as paths and disjoint unions of stars or cliques. Combined with known results, these results settle the parameterized complexity for most of the well-studied structural parameters. We also study the setting where the sets to be reconfigured are small, i.e., their size is part of the parameter, and show that even in this setting the problem is hard for caterpillars, whereas it becomes tractable even for monadic second-order logic when parameterized additionally by shrub-depth.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases Combinatorial Reconfiguration, Token Jumping, Algorithmic Meta-Theorem, Fixed-Parameter Tractability

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.55

Funding This work is partially supported by ANR project ANR-21-CE48-0022 (S-EX-AP-PE-AL).
Tatsuya Gima: JSPS KAKENHI Grant Numbers JP24K23847, JP25K03077.
Yota Otachi: JSPS KAKENHI Grant Numbers JP22H00513, JP24H00697, JP25K03076, JP25K03077.

1 Introduction

Given a base structure (e.g., a graph) and two *feasible* solutions (e.g., independent sets), a combinatorial reconfiguration problem asks if there is a sequence of feasible solutions from one of the given solutions to the other such that every solution in the sequence can be obtained from the one immediately before by a simple operation (e.g., removing a vertex and adding one other). Recently, combinatorial reconfiguration problems have been studied extensively under many different settings of feasibility (such as independent sets, dominating sets, and matchings) and reconfiguration models (token jumping, token sliding, token addition and removal, etc.). See the surveys by van den Heuvel [12], by Nishimura [15], and by Bousquet et al. [4], and the references therein.

We follow the line of work by Mouawad et al. [14] and Gima et al. [10], who gave algorithmic meta-theorems for combinatorial reconfiguration problems on graphs under the token-jumping model, where the feasibility of solutions is expressed by a formula φ in monadic second-order logic (MSO). Mouawad et al. [14] showed that this general problem is fixed-parameter tractable parameterized simultaneously by treewidth, the length of φ, and an upper bound on the length of reconfiguration sequences. Gima et al. [10] worked on the setting where there is no upper bound on the length of reconfiguration sequences and showed that the problem is fixed-parameter tractable parameterized by neighborhood diversity plus the length of φ, and by the sum of treedepth, the length of φ, and the size of the reconfigured sets (which we call the *number of tokens*). On the other hand, they showed that there is a fixed MSO formula φ such that the reconfiguration problem defined by φ is PSPACE-complete even on disjoint unions of trees of treedepth 3.

In this paper, we show intractability results for the problem for extremely restricted settings and thus tightly complement the known positive results. We also present an extension of known fixed-parameter results. Most of our negative results hold for uncolored graphs, while the positive one holds for colored graphs as well. See below for the details.

Our results

Let $\varphi(X)$ be an MSO formula with a free set variable X. (By MSO, we mean MSO_1.) A set $S \subseteq V$ is *φ-feasible* in $G = (V, E)$ if $G \models \varphi(S)$. The problem φ-Reconfiguration asks, given a graph $G = (V, E)$, two φ-feasible sets S and S', whether there is a sequence S_0 $(= S)$, $\ldots$, S_ℓ $(= S')$ of φ-feasible sets such that $|S_i \setminus S_{i+1}| = |S_{i+1} \setminus S_i| = 1$ for $0 \leq i \leq \ell - 1$. The adjacency condition $|S_i \setminus S_{i+1}| = |S_{i+1} \setminus S_i| = 1$ is called the *token jumping* rule. We also study a restricted version of φ-Reconfiguration, where $\varphi(X)$ is a first-order (FO) formula with a free set variable X.

Our results can be split into two groups: one for the general case and one for the case where the number of tokens is part of the parameter.

When the number of tokens is unbounded. In this case, we show PSPACE-completeness of the problem for strongly restricted graph classes.

- **Paths** (Theorem 3.1). There is an FO formula $\varphi(X)$ such that φ-Reconfiguration on uncolored paths is PSPACE-complete.
- **Disjoint unions** (Theorem 3.4). There exists an MSO formula $\varphi(X)$ such that φ-Reconfiguration is PSPACE-complete on disjoint unions of graphs in $\mathcal{C}$, if $\mathcal{C}$ includes, for each order, a polynomial-time constructible connected graph of that order.

These results rule out, under standard complexity assumptions, the possibility of having fixed-parameter algorithms for most of the structural parameters studied in the literature except for vertex integrity and the maximum connected component order. For example, the results above imply that φ-Reconfiguration is PSPACE-complete on graphs of max leaf number 2, graphs of twin-cover number 0, and graphs of distance-to-stars 0.

When the number of tokens is part of the parameter. For this setting, we show that the problem remains intractable (W[1]-hard) even on caterpillars. On the other hand, we show that some known tractability results can be generalized.

- **Caterpillars** (Corollary 4.2). There is an FO formula $\varphi(X)$ with a free set variable X such that φ-Reconfiguration on uncolored caterpillars is W[1]-hard parameterized by the number of tokens. This result is obtained as a corollary to the same hardness result on colored paths (Theorem 4.1).

- **Shrub-depth** (Corollary 4.5). For a graph class with shrub-depth sd, φ-Reconfiguration on it is fixed-parameter tractable parameterized by $\mathsf{sd} + k + |\varphi|$, where k is the number of tokens. This generalizes known results for treedepth and cluster vertex deletion number [10]. We also show that the result holds for any reconfiguration model definable by a fixed MSO formula (Theorem 4.10).

As in the previous setting, these results settle the parameterized complexity for many structural parameters, although some open cases remain. In particular, one of the most important cases, namely uncolored paths, remains unsettled.

2 Preliminaries

Complexity Theory

We use standard notions from computational complexity theory as defined in [2]. For background in parameterized complexity, we refer to [7]. In particular, a *parameterized problem* is a language $L \subseteq \Sigma^*$ together with a polynomial-time computable function $\kappa\colon \Sigma^* \to \mathbb{N}$, the *parameter* of input x. A parameterized problem is *fixed-parameter tractable (fpt)* if it can be decided by a deterministic algorithm with running time $f(k) \cdot n^{O(1)}$ for some computable function $f\colon \mathbb{N} \to \mathbb{N}$, with the usual convention that $k = \kappa(x)$ and $n = |x|$. An *fpt reduction* from a parameterized problem (L, κ) to a parameterized problem (L', κ') is a function $\rho\colon \Sigma^* \to \Sigma^*$ with $x \in L \Leftrightarrow \rho(x) \in L'$ for all $x \in \Sigma^*$ and such that $\kappa'\big(\rho(x)\big) \leq g\big(\kappa(x)\big)$ and ρ is computable in time $f(k) \cdot n^{O(1)}$ for some computable functions f, g. The class W[1] is the class of all parameterized problems that are fpt-reducible to the following problem, which is generally assumed not to be fpt:

p-Clique	
Instance:	Graph $G = (V, E)$, integer $k \geq 1$.
Parameter:	k.
Problem:	Is there a clique of size k in G?

Logic

For background on logic cf. [6, 11]. In particular, we will need first-order logic (FO) and monadic second-order logic (MSO) on (potentially colored) graphs and words: On graphs, first-order variables range over vertices of the graph and atomic formulas are of the form Exy for adjacency and $P_c x$, which states that vertex x has color c. MSO-formulas may additionally quantify over sets of vertices.

On words over an alphabet Σ, first-order variables range over positions of the word and atomic formulas are of the form $\operatorname{succ} xy$ (for "y is the position immediately after x") and $Q_a x$ for "the letter in position x is a", with $a \in \Sigma$. Again, MSO-formulas may additionally quantify over sets of positions; details can be found in Straubing's book [16]. In particular, we need the following result:

▶ **Theorem 2.1** ([16, Thm. IV.3.3]). *A language $L \subseteq \Sigma^*$ is definable in first-order logic if, and only if, it is a locally threshold testable regular language.*

A language $L \subseteq \Sigma^*$ is locally threshold testable if there are numbers $r, t \geq 0$ such that membership of a word $w \in \Sigma^*$ in L is determined by its prefix of length $r-1$, its suffix of length $r-1$, and for each $u \in \Sigma^r$ the number of times it appears as an infix in w, up to threshold t (cf. [16, Sec. IV.3]).

In Sections 3 and 4, we prove hardness results for reconfiguration problems of languages and turn these into hardness results for reconfiguration problems for definable subsets of graphs using *transductions* (also called *interpretations*), cf. [11].

Reconfiguration Problems

We consider various instances of reconfiguration problems. In general, a reconfiguration problem consists of

- a set $\mathcal{S}$ of solutions,
- a symmetric binary relation $E \subseteq \mathcal{S} \times \mathcal{S}$ on the set of solutions.

The task is to decide, given two solutions $s, t \in \mathcal{S}$, whether there is a path between s and t in the graph $(\mathcal{S}, E)$. Specific cases are

φ-Reconfiguration A fixed MSO or FO formula $\varphi(X)$ with a free set variable X defines, in every graph G, a collection

$$\mathcal{S} = \{U \subseteq V(G) \mid G \models \varphi[U]\}$$

of subsets of $V(G)$. Again we set $(A, B) \in E$ if, and only if, $|A \setminus B| = |B \setminus A| = 1$ for $A, B \in \mathcal{S}$. A problem instance consists of a graph G and sets $A, B \in \mathcal{S}$.

L-Reconfiguration For a fixed language $L \subseteq \{0,1\}^*$ over the binary alphabet $\{0,1\}$, we set $\mathcal{S} = L$ and put $(x, y) \in E$ for words $x, y \in L$ if, and only if, $|x| = |y|$ and there are positions $i \neq j$ such that

$$x_i = 1, y_i = 0, \quad x_j = 0, y_j = 1, \quad \text{and } x_k = y_k \text{ for all } k \notin \{i, j\}.$$

In other words, we demand that x and y have the same length and that y be obtained from x by moving one 1 to another position. In particular, x and y have the same number of 1s.

L-TokenReconfiguration For this problem we are given a language $L \subseteq \big(\Sigma \times \{0,1\}\big)^*$ for some finite alphabet Σ. Two words $w \in \Sigma^n$ and $t \in \{0,1\}^n$ are spliced into a word $w \times t$ letter by letter in the natural way, e.g., $ab \times 01 = (a,0)(b,1)$.
Given words $w \in \Sigma^n$ and $s, t \in \{0,1\}^n$, L-TokenReconfiguration asks whether there is a sequence $t_1, \ldots, t_\ell \in \{0,1\}^n$ such that

$$t_1 = s, \quad (w, t_i) \in L \text{ for } 1 \leq i \leq \ell, \text{ and } \quad t_\ell = t$$

such that t_i and t_{i+1} have the same number of 1s and differ in exactly two positions (i.e., one 1 is moved to a different position in going from t_i to t_{i+1}).

Where we consider parameterized variants of these problems, the parameter is the number of elements in $A, B \in \mathcal{S}$ for p-φ-Reconfiguration. For the parameterized problem p-L-TokenReconfiguration, the parameter is the number of 1s in the strings s and t.

3 Hardness with unlimited numbers of tokens

In this section, we show that φ-Reconfiguration is PSPACE-complete even on paths and on disjoint unions of (almost any) graphs. Since φ-Reconfiguration for an MSO-formula φ belongs to PSPACE on general graphs (see [10]), we only show the PSPACE-hardness for the special cases.

3.1 Paths

We prove that the following problem is PSPACE-complete:

▶ **Theorem 3.1.** *There is a first-order formula $\varphi(X)$ with a free set variable X such that φ-Reconfiguration is PSPACE-complete on paths (directed or undirected).*

We slightly reformulate the problem: A path $P = (V, E)$ with n vertices and a set $U \subseteq V$ of vertices of P can be encoded as a string $w \in \{0,1\}^n$ by going along the path from one end to the other and writing a 0 for all positions that are not in U and a 1 for those that are in U. For undirected paths P there are two possible words arising in this way, but the sets U of vertices will, in the cases relevant to us, be such that we can distinguish w from its reverse in first-order logic, so it is clear which of the two strings we mean. Thus, we show that L-Reconfiguration is PSPACE-complete for some locally threshold testable regular language $L \subseteq \{0,1\}^*$.

We reduce the following problem, shown to be PSPACE-complete by Wrochna in [17, Lemma 2.2], to L-Reconfiguration for a suitably chosen language $L \subseteq \{0,1\}^*$:

▶ **Lemma 3.2.** *There is a finite alphabet Σ and a string rewriting system whose word problem is PSPACE-complete and that only contains rules of the form $a_1b_1 \leftrightarrow a_2b_2$ with $a_1 = a_2$ or $b_1 = b_2$.*

String rewriting systems of the form stated in Lemma 3.2 are called 2-*balanced* because each rule replaces a substring of length 2 by one of the same length, and *symmetric* because each rule may be reversed. In addition to these two properties, we are guaranteed that each rule changes only a single one of two consecutive letters.

▶ **Lemma 3.3.** *There is a locally threshold testable language $L \subseteq \{0,1\}^*$ such that L-Reconfiguration is PSPACE-complete.*

Proof. Every locally threshold testable L is in PSPACE (even AC^0), so L-Reconfiguration for such languages can be done in NPSPACE by guessing the correct reconfiguration steps, and therefore it is in PSPACE by Savitch's Theorem. To prove hardness, we reduce the string rewriting problem of Lemma 3.2 to L-Reconfiguration for a language that we will specify. Assume that the alphabet $\Sigma = \{\sigma_1, \dots, \sigma_k\}$ of the rewriting system has k elements and that there are m rules $S_1, \dots, S_m$ of the form

$$S_i \colon \sigma_{\ell_i}\sigma_{r_i} \leftrightarrow \sigma_{\ell'_i}\sigma_{r'_i}.$$

For natural numbers $1 \le a \le b$ we set

$$e_a^{(b)} := 0^{a-1}10^{b-a} \in \{0,1\}^b \qquad \text{and} \qquad r^{(b)} := \left(e_1^{(b)} + \cdots + e_b^{(b)}\right)$$

i.e., $e_a^{(b)}$ is a string of length b with exactly one 1 in its a-th position, and $r^{(b)}$ is a regular expression matching exactly these strings. We define an FO-definable regular language L as

$$L := L_{\text{block}} \cap L_{\text{onerule}} \cap \bigcap_{i=1}^{m} L_i$$

with L_{block} being defined by the regular expression

$$\left(1^3 00 r^{(k)} 001^4 00 r^{(k)} 001^5 00 r^{(m+1)} 00\right)^* 1^3 00 r^{(k)} 001^4 00 r^{(k)} 001^5$$

While this language is defined using a regular expression with Kleene-star, it is easily seen to be locally threshold testable: A word $x \in \{0,1\}^*$ is in L_{block} if, and only if,

- it starts with $1^3 00 r^{(k)} 001^4 0$,
- every infix v of x of length $2k + m + 31$ that starts with $01^4 0$ is of the form

$$01^4 00 r^{(k)} 001^5 00 r^{(m+1)} 001^3 00 r^{(k)} 001^4 0,$$

- and it ends with $01^4 00 r^{(k)} 001^5 00 r^{(m+1)} 001^3 00 r^{(k)} 001^4 00 r^{(k)} 001^5$.

Every word in $L \subseteq L_{\text{block}}$ can be seen as a sequence

$$a_1 b_1 c_1 a_2 b_2 c_2 \dots c_{n-1} a_n b_n$$

of integers with $1 \leq a_j, b_j \leq k$ and $1 \leq c_j \leq m + 1$, and these numbers and whether they are a's, b's or c's can be recovered in FO. Moreover, a single reconfiguration step (i.e., moving a letter 1 to a different position) can only change the value of exactly one number in the sequence. We define the languages L_{onerule} and L_i for $i = 1, \dots, m$ by

- L_{onerule} contains all words for which $c_j = 1$ for all but at most one j and such that if $a_j b_j c_j a_{j+1} b_{j+1}$ is a subsequence with $c_j = 1$ then $a_j = b_j$ and $a_{j+1} = b_{j+1}$.
- If S_i is the rule $\sigma_\ell \sigma_r \leftrightarrow \sigma_{\ell'} \sigma_{r'}$ then L_i contains all words for which, if $a_j b_j c_j a_{j+1} b_{j+1}$ is a subsequence with $c_j = i + 1$, then

$$a_j, b_j \in \{\ell, \ell'\} \quad \text{and} \quad a_{j+1}, b_{j+1} \in \{r, r'\}.$$

The language L only depends on the string rewriting system that we want to reduce from. In particular, k and m are fixed, which is essential for local threshold testability of L.

Our reduction now turns a pair $v, w \in \Sigma^*$ of words of length n, say $v = \sigma_{\alpha_1} \sigma_{\alpha_2} \dots \sigma_{\alpha_n}$ and $w = \sigma_{\beta_1} \sigma_{\beta_2} \dots \sigma_{\beta_n}$, into strings encoding sequences

$$x := \alpha_1 \alpha_1 1 \alpha_2 \alpha_2 1 \cdots \alpha_{n-1} \alpha_{n-1} 1 \alpha_n \alpha_n \quad \text{and} \quad y := \beta_1 \beta_1 1 \beta_2 \beta_2 1 \cdots \beta_{n-1} \beta_{n-1} 1 \beta_n \beta_n.$$

By our choice of L, every rewriting step can be carried out by four reconfiguration steps: Suppose we want to apply a rule $S_i \colon \sigma_\ell \sigma_r \leftrightarrow \sigma_{\ell'} \sigma_r$ to the letters in position j and $j + 1$. We may assume that all c's are equal to 1, and that $a_j = b_j = \ell$ and $a_{j+1} = b_{j+1} = r$ so we can apply the rule from left to right. We now

1. change c_j to $i + 1$,
2. change a_j from ℓ to ℓ'
3. change b_j from ℓ to ℓ'
4. change c_j to 1.

Similarly, one easily checks that every reconfiguration sequence from x to y can be turned into a rewriting sequence from v to w. ◀

Using FO-transductions, we can now prove Theorem 3.1:

Proof of Theorem 3.1. Let L be the language constructed in the proof of Lemma 3.3. Notice that a word $w = w_1 \dots w_n \in \{0, 1\}^n$ can be seen as a directed path on n vertices $v_1, \dots, v_n$ together with a subset $X = \{v_i \mid w_i = 1\}$. Moreover, every word in L (even every word in L_{block}) can be distinguished from its reverse by looking at infixes: There is some constant ℓ such that every word $y \in \{0, 1\}^\ell$ can only be an infix of a word $w \in L$ or its reverse w^{rev}, but not both. With this one can easily define an FO-transduction from 2-colored paths (i.e., paths with a given subset of vertices) to $\{0, 1\}$-words such that a path is valid if, and only if, the interpreted word is in L. ◀

3.2 Disjoint unions

To prove the hardness on disjoint unions (Theorem 3.4), we introduce the source problem 3-Move Subset Sum Reconfiguration [5].

Let $\{a_i \mid i \in [n]\}$ be a set of integers. A sequence $\langle A_0, A_1, \ldots, A_\ell \rangle$ of subsets of $[n]$ is a 3-*move reconfiguration sequence from* A *to* A' if the following conditions hold for all $j \in [\ell]$:

- $\sum_{i \in A_{j-1}} a_i = \sum_{i \in A_j} a_i$;
- $|A_{j-1} \setminus A_j| + |A_j \setminus A_{j-1}| = 3$.

That is, in a 3-move reconfiguration sequence, the sum of the elements corresponding to the sets does not change and the symmetric difference of two consecutive sets has size 3. Note that these conditions imply that there exist distinct indices $x, y, z \in [n]$ such that $a_x + a_y = a_z$ and either

- $A_{j-1} \setminus A_j = \{x, y\}$ and $A_j \setminus A_{j-1} = \{z\}$, or
- $A_{j-1} \setminus A_j = \{z\}$ and $A_j \setminus A_{j-1} = \{x, y\}$.

We call one step in a 3-move reconfiguration sequence a 3-*move*. Cardinal et al. [5] showed that the following problem is PSPACE-complete.

Problem. 3-Move Subset Sum Reconfiguration
Input. A set $\{a_i \mid i \in [n]\}$ of unary-encoded positive integers and sets $A, A' \subseteq [n]$ such that $\sum_{i \in A} a_i = \sum_{i \in A'} a_i$.
Question. Is there a 3-move reconfiguration sequence from A to A'?

Note that in an instance of 3-Move Subset Sum Reconfiguration, $a_i \neq a_j$ for any $i, j \in [n]$ with $i \neq j$ (as $\{a_i \mid i \in [n]\}$ is a set). We use this property in the proof.

We prove the following theorem by presenting a reduction from 3-Move Subset Sum Reconfiguration.

▶ **Theorem 3.4.** *There exists an* MSO *formula* $\varphi(X)$ *with a free set variable* X *such that* φ*-Reconfiguration is PSPACE-complete on disjoint unions of graphs in* $\mathcal{C}$*, where* $\mathcal{C}$ *is any class of (uncolored) graphs such that for every* $p \in \mathbb{Z}^+$*,* $\mathcal{C}$ *includes a connected graph with* p *vertices that can be computed in time polynomial in* p*.*

Furthermore, if every graph in $\mathcal{C}$ *has diameter at most* d*, then* $\varphi(X)$ *can be chosen to be an* FO *formula with a free set variable* X *whose length depends on* d*.*

Proof. Let $\mathcal{I} = \langle \{a_i \mid i \in [n]\}, A, A' \rangle$ be an instance of 3-Move Subset Sum Reconfiguration. For technical reasons, we assume that $a_i \geq 12$ for every $i \in [n]$, which may be obtained by multiplying each a_i by 12 if necessary. From $\mathcal{I}$, we construct a formula φ and an instance $\mathcal{J} = \langle G, S, S' \rangle$ of φ-Reconfiguration as described below.

Construction of $\mathcal{J}$. We construct $\langle G, S, S' \rangle$ as follows:

- for $i \in [n]$, construct a connected graph $C_i \in \mathcal{C}$ with a_i vertices;
- let G be the disjoint union of all C_i for $i \in [n]$;
- set $S = \bigcup_{i \in A} V(C_i)$ and $S' = \bigcup_{i \in A'} V(C_i)$.

Before we formally define φ, let us first explain its intuitive meaning. By taking the vertices of C_i, we represent membership of i in the current subset maintained in 3-Move Subset Sum Reconfiguration. We simulate a 3-move removing x, y and adding z with $a_x + a_y = a_z$ as follows:

1. Perform the following two-move pattern (M) twice: (M) Move one vertex of C_x to C_z, then move one vertex of C_y to C_z.
2. Move all but two remaining vertices of C_x to C_z.
3. Move all but two remaining vertices of C_y to C_z.
4. Perform (M) twice.

We will define φ in such a way that it allows only sets that can appear in the process above (allowing x and y to be swapped at some point in the process). The crucial trick here is that step (M) "marks" the three connected components involved in the current 3-move, and doing it twice forces us to complete all steps without touching other connected components unless we revert all steps executed for the current 3-move.

To define the formula φ, we need to introduce some terms that describe a vertex set. For a set $X \subseteq V(G)$, we say that a component C_i is

- *full* if $V(C_i) \subseteq X$; *empty* if $V(C_i) \cap X = \emptyset$;
- *clean* if it is full or empty; *active* if it is not clean; and
- *occupied* if $|V(C_i) \cap X| \geq |V(C_i)| - 4$.

For a set $X \subseteq V(G)$ and $c \in \mathbb{Z}^+$, we say that an active component C_i is of state

- $+c$ if $|V(C_i) \cap X| = c$; $-c$ if $|V(C_i) \cap X| = |V(C_i)| - c$; and
- $*_c$ if $c \leq |V(C_i) \cap X| \leq |V(C_i)| - c$.

Note that the state of a component can be described in multiple ways (e.g., $+1$ and $-(|V(C_i)| - 1)$), but this causes no problem in what follows.

Using this notation, we describe a set $X \subseteq V(G)$ by the multiset of the states of active components. For example, if X has two components in state -1, one in state $+2$, and the others clean, then we describe X as $\{-1, -1, +2\}$.

We define the formula φ in such a way that $G \models \varphi(X)$ if and only if (1) the number of active components in X is 0, 2, or 3 and (2) the states of the active components can be described as

- $\emptyset$ if there is no active component,
- $\{-1, +1\}$ if there are exactly two active components, and
- one of the following if there are exactly three active components:
  $\{-1, -1, +2\}$, $\{-1, -2, +3\}$, $\{-2, -2, +4\}$, $\{-2, *_3, *_5\}$, $\{-2, +2, *_6\}$,
  $\{+1, +1, -2\}$, $\{+1, +2, -3\}$, $\{+2, +2, -4\}$, $\{+2, *_3, *_5\}$.

▷ Claim 3.5. φ can be expressed in MSO. Furthermore, if each graph in $\mathcal{C}$ has diameter at most d, φ can be expressed in FO with length depending on d.

Proof. An FO-formula with length depending on d can check if two vertices are at distance at most d, and thus, in the bounded-diameter case of the claim, we can express the reachability query. Expressing the reachability query in MSO for general graphs is straightforward.

Using the reachability query, we can check, for each relevant constant c, whether exactly c vertices in the connected component containing v are missing from or included in X. This allows us to decide whether the state of the component having v is one of the allowed states above, and if so, compute the state itself.

Now we express φ as the disjunction of the three cases (having 0, 2, or 3 active components). In the case with i active components, we pick i vertices, express that they are in different connected components, ask that the corresponding active components satisfy the requirement for their states, and check that all other connected components are clean. ◁

Observe that if X satisfies φ, then X admits exactly one of the multisets above as its description since each C_i has at least 12 vertices. Thus, we can call a description of such a set its *state*. If X has state S, we write $X \stackrel{\text{s}}{=} S$. If X has state $\{-2, *_3, *_5\}$, $\{+2, *_3, *_5\}$, or $\{-2, +2, *_6\}$, then we say that X is *unstable*; otherwise, we say that X is *stable*.

Let $X, X' \subseteq V(G)$ be sets with $|X \setminus X'| = |X' \setminus X| = 1$. To simplify the description, we write $X \xrightarrow{-x+y} X'$ when X' is obtained from X by removing a vertex of C_x and adding a vertex of C_y.

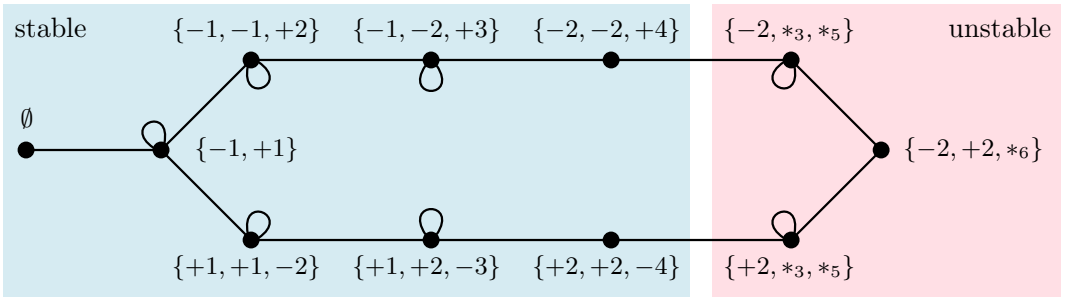

Figure 1 The state-transition of φ-feasible sets. Edges are undirected and some self-loops exist.

The following properties follow by a tedious but straightforward case analysis.

▷ Claim 3.6. ($\star$) If $X, X' \subseteq V(G)$ are φ-feasible and $|X \setminus X'| = |X' \setminus X| = 1$, then:

1. the states of X and X' are adjacent in the state-transition map in Figure 1;
2. if one of X and X' is unstable, then they have the same set of active components;
3. if both X and X' are stable, then they have the same set of occupied components.

In the following, we show that $\mathcal{I} = \langle \{a_i \mid i \in [n]\}, A, A' \rangle$ is a yes-instance of 3-Move Subset Sum Reconfiguration if and only if $\mathcal{J} = \langle G, S, S' \rangle$ is a yes-instance of φ-Reconfiguration.

The only-if direction. Assume that $\mathcal{I}$ is a yes-instance of 3-Move Subset Sum Reconfiguration. Let $\langle A_0, A_1, \ldots, A_\ell \rangle$ be a 3-move reconfiguration sequence from $A_0 = A$ to $A_\ell = A'$. For $0 \leq j \leq \ell$, let $S_j = \bigcup_{i \in A_j} V(C_i)$. Note that $S_0 = S$, $S_\ell = S'$, and each S_j has state $\emptyset$. Hence, it suffices to show that for $j \in [\ell]$, there is a $\mathsf{TJ}(\varphi)$-sequence from S_{j-1} to S_j.

Let $j \in [\ell]$. By symmetry, we may assume that $A_{j-1} \setminus A_j = \{x, y\}$ and $A_j \setminus A_{j-1} = \{z\}$, and thus $a_x + a_y = a_z$. We construct a $\mathsf{TJ}(\varphi)$-sequence $\langle T_0, \ldots, T_{a_z} \rangle$ from $T_0 = S_{j-1}$ to $T_{a_z} = S_j$. Note that each step in this sequence removes one vertex of C_x or C_y and adds one vertex of C_z; that is, either $T_{h-1} \xrightarrow{-x+z} T_h$ or $T_{h-1} \xrightarrow{-y+z} T_h$ holds. Now, we define the sets $T_0, \ldots, T_{a_z}$ as follows.

1. $T_0 \xrightarrow{-x+z} T_1 \xrightarrow{-y+z} T_2 \xrightarrow{-x+z} T_3 \xrightarrow{-y+z} T_4$;
   - the state changes as $\emptyset \to \{-1,+1\} \to \{-1,-1,+2\} \to \{-1,-2,+3\} \to \{-2,-2,+4\}$;
   - $|T_4 \cap V(C_x)| = |V(C_x)| - 2$, $|T_4 \cap V(C_y)| = |V(C_y)| - 2$, and $|T_4 \cap V(C_z)| = 4$.
2. $T_{h-1} \xrightarrow{-x+z} T_h$ for $5 \leq h \leq a_x$;
   - $T_h \overset{\mathrm{s}}{=} \{-2, *_3, *_5\}$ for $5 \leq h \leq a_x - 1$;
   - $T_{a_x} \overset{\mathrm{s}}{=} \{-2, +2, +a_x\}$, which is $\{-2, +2, *_6\}$ as $6 \leq a_x = a_z - a_y \leq a_z - 6$;
   - $|T_{a_x} \cap V(C_x)| = 2$, $|T_{a_x} \cap V(C_y)| = |V(C_y)| - 2$, and $|T_{a_x} \cap V(C_z)| = a_x$.
3. $T_{h-1} \xrightarrow{-y+z} T_h$ for $a_x + 1 \leq h \leq a_z - 4$ $(= a_x + a_y - 4)$;
   - $T_h \overset{\mathrm{s}}{=} \{+2, *_3, *_5\}$ for $a_x + 1 \leq h \leq a_z - 5$;
   - $T_{a_z-4} \overset{\mathrm{s}}{=} \{+2, +2, -4\}$;
   - $|T_{a_z-4} \cap V(C_x)| = 2$, $|T_{a_z-4} \cap V(C_y)| = 2$, and $|T_{a_z-4} \cap V(C_z)| = a_z - 4$.
4. $T_{a_z-4} \xrightarrow{-x+z} T_{a_z-3} \xrightarrow{-y+z} T_{a_z-2} \xrightarrow{-x+z} T_{a_z-1} \xrightarrow{-y+z} T_{a_z}$.
   - the state changes as $\{+2,+2,-4\} \to \{+1,+2,-3\} \to \{+1,+1,-2\} \to \{+1,-1\} \to \emptyset$;
   - $|T_{a_z} \cap V(C_x)| = 0$, $|T_{a_z} \cap V(C_y)| = 0$, and $|T_{a_z} \cap V(C_z)| = a_z$.

Clearly, $T_{a_z} = S_j$.

The if direction. Assume that $\mathcal{J}$ is a yes-instance of φ-Reconfiguration. Let $\langle S_0, \ldots, S_\ell \rangle$ be a $\mathsf{TJ}(\varphi)$-sequence from $S_0 = S$ to $S_\ell = S'$. For each $i \in \{0, \ldots, \ell\}$, if S_i is stable, then we define $A_i = \{j \in [n] \mid C_j \text{ is occupied in } S_i\}$. Note that the sets $S_0 = S$ and $S_\ell = S'$ are stable, and we have $A_0 = A$ and $A_\ell = A'$. Note also that by Claim 3.6 (3), if S_i and S_{i+1} are stable, then $A_i = A_{i+1}$ holds.

Let $p, q \in \{0, \ldots, \ell\}$ be any indices such that $p < q$, S_p and S_q are stable, and every S_r with $p < r < q$ is unstable.

▷ Claim 3.7. If $A_p \neq A_q$, then $\sum_{i \in A_p} a_i = \sum_{i \in A_q} a_i$ and $|A_p \setminus A_q| + |A_q \setminus A_p| = 3$.

Proof. Since S_p is stable and S_{p+1} is unstable, Claim 3.6 (1) implies that the state of S_p is either $\{-2, -2, +4\}$ or $\{+2, +2, -4\}$. Similarly, the state of S_q is either $\{-2, -2, +4\}$ or $\{+2, +2, -4\}$. Here we assume that $S_p \overset{\mathrm{s}}{=} \{-2, -2, +4\}$. (The other case $S_p \overset{\mathrm{s}}{=} \{+2, +2, -4\}$ is symmetric.) Let $x, y, z \in [n]$ be the indices such that C_x and C_y are -2 and C_z is $+4$ in S_p. If $S_q \overset{\mathrm{s}}{=} \{-2, -2, +4\}$ as well, then C_z has to be -2 in S_q as $A_p \neq A_q$. By symmetry, assume that C_x is -2 and C_y is $+4$ in S_q. By Claim 3.6 (2), the sequence from S_p to S_q involves changes only in C_x, C_y, and C_z, and hence, it holds that $(|V(C_x)| - 2) + (|V(C_y)| - 2) + 4 = (|V(C_x)| - 2) + 4 + (|V(C_z)| - 2)$, which implies $|V(C_y)| = |V(C_z)|$, a contradiction. (Recall that $a_y \neq a_z$.) Thus we conclude that $S_q \overset{\mathrm{s}}{=} \{+2, +2, -4\}$. Suppose to the contrary that C_z is $+2$ in S_q. By symmetry, assume that C_x is $+2$ and C_y is -4 in S_q. Then we have $(|V(C_x)| - 2) + (|V(C_y)| - 2) + 4 = 2 + (|V(C_y)| - 4) + 2$, which implies $|V(C_x)| = 0$, a contradiction. Hence, C_x and C_y are $+2$ and C_z is -4 in S_q. This implies that $A_p \setminus A_q = \{x, y\}$ and $A_q \setminus A_p = \{z\}$. Since $(|V(C_x)| - 2) + (|V(C_y)| - 2) + 4 = 2 + 2 + (|V(C_z)| - 4)$, we have $|V(C_x)| + |V(C_y)| = |V(C_z)|$, which implies $a_x + a_y = a_z$ and $\sum_{i \in A_p} a_i = \sum_{i \in A_q} a_i$. ◁

Let $\ell' + 1$ be the number of stable sets in $\langle S_0, \ldots, S_\ell \rangle$. For $0 \leq j \leq \ell'$, let $f(j)$ be the jth index such that $S_{f(j)}$ is stable. Note that $f(0) = 0$ and $f(\ell') = \ell$. By Claim 3.7, for $j \in [\ell']$, if $A_{f(j-1)} \neq A_{f(j)}$, then $\sum_{i \in A_{f(j-1)}} a_i = \sum_{i \in A_{f(j)}} a_i$ and $|A_{f(j-1)} \setminus A_{f(j)}| + |A_{f(j)} \setminus A_{f(j-1)}| = 3$. Therefore, by deleting consecutive duplicates in the sequence $\langle A_{f(0)}, \ldots, A_{f(\ell')} \rangle$, we obtain a 3-move reconfiguration sequence from $A_{f(0)}$ $(= A_0 = A)$ to $A_{f(\ell')}$ $(= A_\ell = A')$. ◀

4 The number of tokens as a parameter

Notice that the instances of φ-Reconfiguration that our reduction in the proof of Theorem 3.1 produces have large source and target sets, namely of size $\Omega(n)$ when n is the length of the path. On the other hand, there are only $O(n^k)$ sets of k vertices in a graph with n vertices, so φ-Reconfiguration can be done in time $O(n^{|s|+c})$ if the source (and target) set has $|s|$ elements and model checking for φ can be done in time $O(n^c)$. In parameterized complexity, this puts the problem into the class XP.

It remains open whether the parameterized variant p-φ-Reconfiguration on paths, or equivalently p-L-Reconfiguration for regular languages $L \subseteq \{0, 1\}^*$, is fixed-parameter tractable. We do get W[1]-hardness for φ-Reconfiguration on caterpillar graphs parameterized by the size of the source set. On the other hand, when parameterized by $\mathrm{sd} + |s| + |\varphi|$, where sd is the shrub-depth of the input graph, even MSO-Reconfiguration is fixed-parameter tractable.

4.1 Hardness on colored paths and caterpillars

▶ **Theorem 4.1.** *There is a first-order formula $\varphi(X)$ with a free set variable X such that the reconfiguration problem for φ on colored paths is* W[1]*-hard parameterized by $|X|$.*

Proof. We prove hardness of p-L-TokenReconfiguration for a suitably defined locally threshold testable language L by reducing the W[1]-complete problem p-Clique to it. Hardness of p-φ-Reconfiguration on colored paths then follows by reducing p-L-TokenReconfiguration to it, since words over finite alphabets can be FO-interpreted in colored paths (note that we only need to define the successor relation on positions in the word, and an orientation of the edges of a colored path can be encoded by using three times the original number of colors).

We first note that for p-L-TokenReconfiguration, we may assume that there are r different token types (with r being a constant not depending on the parameter), and a letter may receive one token of each type. This can be done by introducing a new letter to the alphabet and inserting this letter $(r-1)$ times after each letter of the original word. The type of a token then depends on whether it is placed on the original letter or one of the $(r-1)$ letters following it. In principle, tokens may change their type when being moved.

Given the adjacency matrix $A = (a_{ij})_{1 \leq i,j \leq n}$ of a graph, our reduction outputs a string

$$w_A := a^n b x_1 c x_2 c \dots c x_n d d d \in \{a, b, c, d, 0, 1\}^{n^2+2n+3},$$

where $x_i \in \{0,1\}^n$ is the i-th row of $A + E_n$, the adjacency matrix with the diagonal entries replaced by 1s. Using the above mentioned construction, we assume that there are 11 different kinds of tokens named $1, \dots, 9, \dagger, \ddagger$. The intuition behind the reduction is as follows:

- The first n letters of w_A represent the n vertices of the input graph. We encode a subset of k vertices by placing k tokens of type $\dagger$ on the corresponding letters of w_A.
- The subword starting at the letter b encodes the adjacency matrix of the input graph, with letters c encoding "carriage returns".
- In a successful reconfiguration sequence, one first guesses a k-clique using the $\dagger$-tokens and then checks that it is indeed a clique. The $\ddagger$-tokens are used to make sure that the guessed clique cannot be changed during the check. Tokens of type 1 to 9 are used to perform the check.

For the start and target tokenizations, we place a total of $2k + 9$ tokens (where k is the parameter, i.e., the size of the clique; we assume this to be ≥ 3) as follows:

- one token of type $\dagger$ and $\ddagger$ each on the first k letters of the word (all of which are as)
- tokens $1, 2, 3$ and $4, 5, 6$ on the first three letters, so that the first letter receives tokens $\dagger$, $\ddagger$, 1, and 4, the second letter receives $\dagger$, $\ddagger$, 2, and 5, and the third receives $\dagger$, $\ddagger$, 3, and 6,
- tokens $7, 8, 9$ on the three consecutive letters starting with the b in the $(n+1)$-st position.

The target tokenization t is the same except for the tokens $7, 8, 9$, which are placed on the last three letters of w_A in t.

The language L consists of all tokenized words that satisfy conditions L1) to L11) below. In particular, tokens 1 to 9 must appear exactly once by L1), so for $x, y, z \in \{1, \dots, 9\}$ it makes sense to use the following notation and Boolean combinations thereof:

- $d(x, y)$ is the distance between the position of token x and the position of token y,
- $x \prec y$ means that token x appears strictly to the left of token y,
- $x \cdot\cdot y$ means that $d(x, y) = 3$,
- $x\cdot$ means that the token $x{+}1$ is not immediately to the right of token x (i.e., $d(x, x{+}1) > 1$),
- $\cdot x$ means that the token $x{-}1$ is not immediately to the left of token x (i.e., $d(x{-}1, x) > 1$),
- xyz means that tokens x, y, and z appear on consecutive letters,
- α_x for $\alpha \in \{a, b, c, d, 0, 1\}$ means that token x is on a letter α.

With this notation, the conditions that define the language L are:

L1) each of the tokens 1 to 9 appears exactly once

L2) tokens $\dagger$ and $\ddagger$ may only be on as

L3) unless $(b_7 \vee d_7)$, every position with token † must also have a token ‡ (this implies that tokens † and ‡ can only be moved when token 7 is on a b or d)

L4) $\neg a_7$ (token 7 never appears on a letter a)

L5) $1 \prec 2 \prec 3$, $4 \prec 5 \prec 6$, and $7 \prec 8 \prec 9$

L6) $d(1,3), d(4,6), d(7,9) \leq 3$

L7) If $b_7 \vee c_7$ and $7 \cdot\cdot 9$, then tokens 1, 2, and 3 are placed on the first three letters of the word.

L8) If $\neg(c_7 \vee b_7 \vee d_7)$ then $789 \Rightarrow 12, 8\cdot \Rightarrow 1\cdot\cdot 3$, and $\cdot 8 \Rightarrow 23$.

L9) If $\neg(c_7 \vee d_7)$ then $d(4,6) = 2$

L10) If c_7 then $789 \Rightarrow 45, 8\cdot \Rightarrow 4\cdot\cdot 6$, and $\cdot 8 \Rightarrow 56$.

L11) If $d(1,3) = d(4,6) = d(7,9) = 2$ and both 1 and 4 are on a letter that also has token †, then $\neg 0_7$.

Note that by L6), condition L5) is locally testable, so all of these conditions are locally threshold testable.

Conditions L6) and L5) together imply that the tokens 123 can only be "slid" along the word: The only way to move them is to first move 3 one position to the right, followed by 2 and then 1, or the other direction (first 1 one step to the left, then 2, then 3). The same applies to 456 and 789. The start and target tokenizations s and t only differ in the position of tokens 789, so these must be slid all along the word from their starting position to the end.

Before starting to slide 789, the † and ‡-tokens may be placed on k of the letters in the a^n-prefix of w_A. Once we start sliding 789, these tokens must remain where they are by L3), until 789 has reached the end of the word. In this way the †-tokens can be used to "guess" a clique of size k by placing them on the k positions in the a^n-prefix of w_A that correspond to the vertices of the clique.

Since 7 is never on a letter a by L4), we can only move 789 one step to the right: 456 is fixed by L9) and 123 might be slid, but by L7) it has to be returned to the beginning of the word before we can start sliding 789.

Condition L8) implies that 123 and 789 must be moved synchronously now: Before 789 can be moved further, we must first move 3 one to the right, then 9, then 2, then 8, then 1, then 7. This continues until 7 is on the first letter c: Now 123 can be slid freely, and before we move 789 further to the right, 123 must be slid back to the beginning of the word by L7). Furthermore, 456 must be slid one step to the right together with 789 by L10). Thus the c-letters in w_A serve as "carriage returns".

Finally, if $d(1,3) = d(4,6) = d(7,9) = 2$ and tokens 1 and 4 are on positions j and i of the word, then token 7 is on the letter corresponding to the entry in the i-th row and j-th column of $A + E_n$ or on a letter b, c, or d. Therefore condition L11) makes sure that every time 123 and 456 are on letters marked with † (i.e., vertices in the guessed clique), there is an edge or a loop between these vertices in the graph.

Again, a word $w \in \Sigma^*$ can be encoded as a colored path. In order to be able to retrieve the direction of the path, we use colors $\Sigma \times \{0,1,2\}$ and turn a word $x_1x_2x_3 \dots x_n \in \Sigma^n$ into a path with colors $(x_1,1)(x_2,2)(x_3,0)(x_4,1)\cdots(x_n,(n \bmod 3))$, so it is clear which of the two neighbors of a vertex is its predecessor and which is the successor. ◀

We can now use FO-interpretations (cf. [11]) to interpret a colored path in an uncolored caterpillar graph to obtain the following corollary:

▶ **Corollary 4.2.** *There is a first-order formula $\varphi(X)$ with a free set variable X such that the reconfiguration problem for φ on caterpillars is* W[1]*-hard parameterized by $|X|$.*

Bodlaender et al. recently showed that clique reconfiguration (on arbitrary finite graphs) is XL-complete when parameterized by the number of vertices [3]. The technique used in our proof of Theorem 4.1 can be used to obtain a reduction from parameterized clique reconfiguration to p-L-TokenReconfiguration for some locally threshold testable language L, strengthening both Theorem 4.1 and Corollary 4.2 from W[1]-hardness to XL-completeness. Details will appear in the full version of this paper.

4.2 Shrub-depth: an extension of known FPT cases

Gima et al. [10] showed that φ-Reconfiguration is fixed-parameter tractable parameterized by treedepth $+ k + |\varphi|$, or by cluster vertex deletion number $+ k + |\varphi|$, where φ is an MSO formula, k is the solution size, and $|\varphi|$ is the length of φ. In this section, we generalize this result to parameterization by shrub-depth, which is a generalization of treedepth and cluster vertex deletion number.

A *tree-model* with m *labels* and *depth* d for a graph G is a rooted tree T such that

1. the set of leaves of T is exactly the set of vertices of G;
2. each leaf in T has distance exactly d to the root;
3. each leaf in T has one label from $[m]$; and
4. for any two vertices u, v of G, the existence of the edge $\{u, v\}$ in G depends only on the labels of u and v and the distance between u and v in T.

Let $\mathcal{TM}_m(d)$ denote the class of graphs admitting a tree-model with m labels and depth d. The *shrub-depth* of a class of graphs $\mathcal{C}$, denoted $\mathsf{sd}(\mathcal{C})$, is the minimum integer d such that $\mathcal{C} \subseteq \mathcal{TM}_m(d)$ for some integer m. Note that shrub-depth is defined for a class of graphs, not for a single graph, since every graph G admits a tree-model with $|V(G)|$ labels and depth 1.

To see the relation between shrub-depth and other parameters, we recall the definitions of treedepth and cluster vertex deletion number. The *treedepth* of a graph G is the minimum depth of a rooted forest F such that G is a subgraph of the closure of F, where the closure of a forest F is the graph obtained from F by adding an edge between every ancestor-descendant pair in F. The *cluster vertex deletion number* of a graph G is the minimum size of a vertex set S such that $G - S$ is a cluster graph, where a cluster graph is a disjoint union of cliques. The parameter shrub-depth generalizes the above two parameters in the following sense. If a graph G has treedepth at most d, then $G \in \mathcal{TM}_{2^d}(d)$ [8], and if a graph G has cluster vertex deletion number at most d, then $G \in \mathcal{TM}_{2^d+d}(2)$. The latter is a straightforward consequence from the definition of tree-model.

▶ **Observation 4.3** (folklore). ($\star$) *Let G be a graph with cluster vertex deletion number at most d. Then $G \in \mathcal{TM}_{2^d+d}(2)$.*

Let G be a graph, φ be a formula, and A, B be vertex sets of G. We denote by $\mathrm{dist}_{\varphi,G}(A, B)$ the length of a shortest $\mathsf{TJ}(\varphi)$-sequence from A to B.

We show that MSO-Reconfiguration admits a "distance-preserving kernel" on a class $\mathcal{C}$ parameterized by $\langle$the shrub-depth of $\mathcal{C}\rangle + k + |\varphi|$, in the following sense.

▶ **Theorem 4.4.** *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-Reconfiguration *with $|S| = |S'| = k$. Let $m, d \in \mathbb{N}$. Given a tree-model T_G of G with m labels and depth d, we can compute in time $|V(G)|^{\mathcal{O}(1)}$ a subgraph H of G such that $\mathrm{dist}_{\varphi,G}(S, S') = \mathrm{dist}_{\varphi,H}(S, S')$ and the number of vertices of H depends only on $m + d + k + |\varphi|$.*

▶ **Corollary 4.5.** *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-Reconfiguration *with $|S| = |S'| = k$. Let $m, d \in \mathbb{N}$. If $G \in \mathcal{TM}_m(d)$, then computing the distance $\mathrm{dist}_{\varphi,G}(S, S')$ is fixed-parameter tractable parameterized by $m + d + k + |\varphi|$.*

Proof. Constructing a tree-model of G with m labels and depth d is fixed-parameter tractable parameterized by $m + d$ [9]. Then Theorem 4.4 yields a subgraph H with $\mathrm{dist}_{\varphi,G}(S, S') = \mathrm{dist}_{\varphi,H}(S, S')$ and $|V(H)| = f(m + d + k + |\varphi|)$. We can compute $\mathrm{dist}_{\varphi,H}(S, S')$ in $(2^{|V(H)|})^{O(|\varphi|)}$ time by combining with a brute-force model checking algorithm and a shortest path algorithm, which implies the claim. ◀

We remark that Theorem 4.4 is proved similarly to the proof for treedepth [10] and to kernelization techniques for MSO model checking on bounded shrub-depth graphs [8]. However, we include a full proof for the sake of completeness.

For an MSO formula ψ, let $\mathsf{q}(\psi) = 2^{\mathsf{q}_\mathrm{s}} \cdot \mathsf{q}_\mathrm{v}$, where q_s and q_v are the numbers of set and vertex quantifiers in ψ, respectively.

Let $G = (V, E, \mathcal{C})$ be a colored graph and $X, X' \subseteq V$. The vertex sets X and X' are of the same *type* if there is an automorphism η of G such that $\eta(X) = X'$, $\eta(X') = X$, $\eta(v) = v$ for every $v \notin X \cup X'$, and $\mathcal{C}(v) = \mathcal{C}(\eta(v))$ for every $v \in V$. It is known that if there are a sufficiently large number of disjoint vertex sets of the same type, then we can safely remove some of them [10].

▶ **Lemma 4.6** ([10]). *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-*Reconfiguration with $|S| = |S'| = k$. Let $\{C_1, \ldots, C_t\}$ be a family of disjoint size-p vertex sets of the same type not intersecting $S \cup S'$. If $t > k + (\mathsf{q}(\varphi))^p$, then $\mathrm{dist}_{\varphi,G}(S, S') = \mathrm{dist}_{\varphi,G-C_i}(S, S')$ for every $C_i \in \{C_1, \ldots, C_t\}$.*

The following lemma is immediate from the definition of tree-model and type, but we include a proof for the sake of completeness.

▶ **Lemma 4.7.** ($\star$) *Let G be a graph and T_G be a tree-model of G. Let w be a non-leaf node in T_G. Let T_1 and T_2 be two subtrees of T_G rooted at two children of w, and L_i be the set of leaves of T_i for $i \in \{1, 2\}$. If there is an isomorphism f from T_1 to T_2 that preserves colors (in G) and labels (in T_G) of the leaves, then $L_1 \subseteq V(G)$ and $L_2 \subseteq V(G)$ have the same type in G.*

Proof of Theorem 4.4. We exhaustively apply the following reduction, which is safe by Lemmas 4.6 and 4.7, in a bottom-up manner along T_G. Let v be a non-leaf node in T_G. Let $T_1, \ldots, T_p$ be the subtrees of T_G rooted at children of v and having no intersection with $S \cup S'$. For each pair (T_i, T_j) of subtrees, we check whether there exists an isomorphism from T_i to T_j that preserves colors and labels. This can be done in polynomial time by a labeled-tree isomorphism checking algorithm [1]. If some T_i is color-label preserving isomorphic to $k + (\mathsf{q}(\varphi))^\ell$ or more other subtrees, where ℓ is the number of leaves in T_i, then we remove T_i from T_G and the leaves of T_i from G. This is safe by Lemmas 4.6 and 4.7. We call the obtained graph H and the corresponding tree-model T_H.

The *height* of a node in a rooted tree is the distance to a closest leaf. In the rest of the proof, we show that each node of height h has at most $c(h)$ children and each subtree rooted at a node of height h contains at most $\ell(h)$ leaves, where $c(0) = 0$, $\ell(0) = 1$, and for $h \geq 0$,

$$c(h+1) = (k + (\mathsf{q}(\varphi))^{\ell(h)}) \cdot 4^{h \cdot \ell(h)} \cdot 2^{|\varphi| \cdot \ell(h)} \cdot m^{\ell(h)} + 2k,$$
$$\ell(h+1) = \ell(h) \cdot c(h+1).$$

This implies that H has at most $\ell(d)$ vertices, where $\ell(d)$ depends only on m, d, k, and $|\varphi|$.

The claim is trivial when $h = 0$. Assume that the claim holds for some $h \geq 0$. We only prove the upper bound $c(h + 1)$ for the number of children as the upper bound $\ell(h + 1)$ follows immediately.

Let v be a node of height $h+1$. Among the subtrees rooted at the children of v, let $T_1, \ldots, T_p$ be the ones having no intersection with $S \cup S'$. Note that v has at most $p+2k$ children. We denote by L_i the set of leaves in T_i for $i \in [p]$.

We define an equivalence relation $\simeq$ on $\{T_1, \ldots, T_p\}$ in such a way that $T_i \simeq T_j$ if and only if there is an isomorphism from T_i to T_j that preserves colors and labels of the leaves. The relation partitions $\{T_1, \ldots, T_p\}$ into at most $4^{h \cdot \ell(h)} \cdot 2^{|\varphi| \cdot \ell(h)} \cdot m^{\ell(h)}$ equivalence classes, where $4^{h \cdot \ell(h)}$ upper-bounds the number of rooted trees of depth h and $\ell(h)$ leaves, $2^{|\varphi| \cdot \ell(h)}$ upper-bounds the number of possible ways for coloring $\ell(h)$ vertices with subsets of at most $|\varphi|$ colors, and $m^{\ell(h)}$ upper-bounds the number of possible ways for labeling $\ell(h)$ vertices with m labels. We can see that each equivalence class has size at most $k + (\mathsf{q}(\varphi))^{\ell(h)}$ as the reduction was exhaustively applied in a bottom-up manner. Therefore, we can bound the number of children of v as

$$p + 2k \leq (k + (\mathsf{q}(\varphi))^{\ell(h)}) 4^{h \cdot \ell(h)} \cdot 2^{|\varphi| \cdot \ell(h)} \cdot m^{\ell(h)} + 2k = c(h+1). \qquad \blacktriangleleft$$

The proof of Theorem 4.4 does not use any property of the reconfiguration rules, but only the result of Lemma 4.6. Gima et al. [10, Section 6] mentioned that Lemma 4.6 also holds under the token-sliding rule and obtained meta-theorems for the token-sliding rule. We generalize their observation to any "MSO-definable rule" (defined later), which implies that Theorem 4.4 can be generalized to any MSO-definable rule.

4.2.1 Extension to any MSO-definable reconfiguration rule

An MSO-*definable reconfiguration rule* (MSO-*rule*) is an MSO formula $\rho(X, Y)$ with free set variables X, Y. Fix a positive integer k. A *k-bounded reconfiguration sequence* in G under ρ from S to S' is a sequence $S = S_0, \ldots, S_\ell = S'$ with $G \models \rho(S_{i-1}, S_i)$ for all $i \in [\ell]$ and $|S_i| \leq k$ for all $i \in \{0, \ldots, \ell\}$. For an MSO formula $\varphi(X)$, let $\mathrm{dist}^{\rho,k}_{\varphi,G}(S, S')$ be the minimum length of such a sequence with $G \models \varphi(S_i)$ for all $i \in \{0, \ldots, \ell\}$. This naturally generalizes MSO-Reconfiguration.

▶ **Example 4.8.** The *token-jumping (TJ)* rule can be encoded by an MSO-formula $\rho_{\mathsf{TJ}}(X, Y) \equiv \exists x \exists y (x \in X \setminus Y \wedge y \in Y \setminus X \wedge (X \setminus \{x\} = Y \setminus \{y\}))$. The *token-sliding (TS)* rule can be encoded by the MSO-formula obtained from $\rho_{\mathsf{TJ}}(X, Y)$ by adding the condition that x and y are adjacent. The *token addition/removal (TAR)* rule can be encoded by an MSO-formula $\rho_{\mathsf{TAR}}(X, Y) \equiv ((\exists x \notin X (X \cup \{x\} = Y)) \vee (\exists x \in X (X \setminus \{x\} = Y)))$.

We next fix an MSO-rule ρ and threshold k. We extend Lemma 4.6 as follows.

▶ **Lemma 4.9.** *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-Reconfiguration. *Let $\{C_1, \ldots, C_t\}$ be a family of disjoint size-p vertex sets of the same type not intersecting $S \cup S'$. If $t > 2k + (\max(\mathsf{q}(\rho), \mathsf{q}(\varphi)))^p$, then $\mathrm{dist}^{\rho,k}_{\varphi,G}(S, S') = \mathrm{dist}^{\rho,k}_{\varphi,G-C_i}(S, S')$ for every $C_i \in \{C_1, \ldots, C_t\}$.*

Before proving Lemma 4.9, we see that Theorem 4.4 can be generalized to any MSO-definable rule. This is simply done by replacing Lemma 4.6 with Lemma 4.9 and the corresponding parts in the proof of Theorem 4.4. Thus, we obtain the following meta-theorem.

▶ **Theorem 4.10.** *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-Reconfiguration. *Given a tree-model T_G of G with m labels and depth d, we can compute in time $|V(G)|^{\mathcal{O}(1)}$ a subgraph H of G such that $\mathrm{dist}^{\rho,k}_{\varphi,G}(S, S') = \mathrm{dist}^{\rho,k}_{\varphi,H}(S, S')$ and the number of vertices of H depends only on $m + d + k + |\varphi| + |\rho|$.*

In the following, we prove Lemma 4.9. The proof technique is almost the same as that of Lemma 4.6 [10] but we need to carefully handle the reconfiguration rules. The following is a fundamental property of MSO formulas on (colored) graphs.

▶ **Lemma 4.11** (folklore, see e.g., [13]). *Let k be a positive integer and φ be an* MSO *formula with k free set variables $\vec{X}$. Let G be a colored graph and $\vec{S}$ be a tuple of k vertex sets of G. For any automorphism η of G that preserves the colors of vertices, we have $G \models \varphi(\vec{S})$ if and only if $G \models \varphi(\eta(\vec{S}))$.*

The following lemma states that there is a shortest reconfiguration sequence that intersects only a small number of sets in the family $\{C_1, \ldots, C_t\}$ of same-type vertex sets.

▶ **Lemma 4.12.** ($\star$) *Let $\langle \varphi, G, S, S' \rangle$ be an instance of* MSO-Reconfiguration *with $|S|, |S'| \leq k$. Let $\{C_1, \ldots, C_t\}$ be a family of disjoint size-p vertex sets of the same type not intersecting $S \cup S'$. If $t > 2k$, then for every $I \subseteq [t]$ with $|I| = 2k$, there is a shortest k-bounded reconfiguration sequence $S_0, \ldots, S_\ell$ from S to S' under ρ such that $S_i \cap \bigcup_{j \notin I} C_j = \emptyset$ for every $i \in [\ell]$.*

The following lemma is a key ingredient to prove Lemma 4.9.

▶ **Lemma 4.13** ([13]). *Let G be a colored graph and φ be an* MSO *formula with c free set variables $\vec{X}$. Assume that G contains $t > (\mathsf{q}(\varphi))^p$ disjoint size-p vertex sets $C_1, \ldots, C_t$ with the same type. Then, for any c-tuple of vertex sets $\vec{S}$ of G disjoint from the t sets, $G \models \varphi(\vec{S})$ if and only if $G - C_i \models \varphi(\vec{S})$ for any $i \in [t]$.*

Proof of Lemma 4.9. It is sufficient to show that $\mathrm{dist}^{\rho,k}_{\varphi,G}(S, S') = \mathrm{dist}^{\rho,k}_{\varphi,G-C_1}(S, S')$. Recall that $C_1, \ldots, C_t$ are disjoint size-p vertex sets of the same type not intersecting $S \cup S'$ and $t > 2k + (\max(\mathsf{q}(\rho), \mathsf{q}(\varphi)))^p$. By Lemma 4.12, for an index set $I = [2, 2k+1]$, there is a shortest k-bounded reconfiguration sequence $S_0, \ldots, S_\ell$ from S to S' under ρ such that $S_i \cap \bigcup_{j \notin I} C_j = \emptyset$ for every $i \in [\ell]$. Here, $\{C_j \mid j \in [t] \setminus I\} (\ni C_1)$ is a family of $t - 2k > (\max(\mathsf{q}(\rho), \mathsf{q}(\varphi)))^p$ disjoint size-p vertex sets of the same type not intersecting S_i for every $i \in [\ell]$. By Lemma 4.13, for every $i \in [1, \ell]$ we have $G \models \rho(S_{i-1}, S_i)$ iff $G - C_1 \models \rho(S_{i-1}, S_i)$, and for every $i \in \{0, \ldots, \ell\}$ we have $G \models \varphi(S_i)$ iff $G - C_1 \models \varphi(S_i)$. Hence sequences avoiding $\bigcup_{j \notin I} C_j$ are exactly the same valid k-bounded reconfiguration sequences in G and $G - C_1$, and thus $\mathrm{dist}^{\rho,k}_{\varphi,G}(S, S') = \mathrm{dist}^{\rho,k}_{\varphi,G-C_1}(S, S')$. ◀

5 Conclusion

Even though Theorems 3.1 and 3.4 settle the parameterized complexity of φ-Reconfiguration for a lot of the classical structural parameters, some of them are left open; in particular, graphs of bounded vertex integrity or maximum size of a connected component would be interesting cases to study. Similarly, Theorem 4.1 naturally raises the question of the complexity of φ-Reconfiguration parameterized by the number of tokens on uncolored paths.

Finally, it would be interesting to see whether these results transfer to other models. In particular, it seems that the presented constructions yielding hardness results can be adapted for the token sliding model, where a token is only allowed to move from one vertex to one of its neighbors, in the following way: we add a universal vertex, i.e., a vertex that is connected to every other vertex of the graph; that way, two token-sliding steps through this universal vertex can simulate a token-jumping step.

References

1 Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

3 Hans L. Bodlaender, Carla Groenland, and Céline M. F. Swennenhuis. Parameterized complexities of dominating and independent set reconfiguration. *Algorithmica*, 88(3):39, 2026. doi:10.1007/s00453-026-01381-9.

4 Nicolas Bousquet, Amer E. Mouawad, Naomi Nishimura, and Sebastian Siebertz. A survey on the parameterized complexity of reconfiguration problems. *Comput. Sci. Rev.*, 53:100663, 2024. doi:10.1016/J.COSREV.2024.100663.

5 Jean Cardinal, Erik D. Demaine, David Eppstein, Robert A. Hearn, and Andrew Winslow. Reconfiguration of satisfying assignments and subset sums: Easy to find, hard to connect. *Theor. Comput. Sci.*, 806:332–343, 2020. doi:10.1016/j.tcs.2019.05.028.

6 Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer, 1999.

7 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Springer, 2006. doi:10.1007/3-540-29953-X.

8 Jakub Gajarský and Petr Hliněný. Kernelizing MSO properties of trees of fixed height, and some consequences. *Log. Methods Comput. Sci.*, 11(1), 2015. doi:10.2168/LMCS-11(1:19)2015.

9 Jakub Gajarský and Stephan Kreutzer. Computing shrub-depth decompositions. In *STACS 2020*, LIPIcs, pages 56:1–56:17, 2020. doi:10.4230/LIPIcs.STACS.2020.56.

10 Tatsuya Gima, Takehiro Ito, Yasuaki Kobayashi, and Yota Otachi. Algorithmic meta-theorems for combinatorial reconfiguration revisited. *Algorithmica*, 86(11):3395–3424, 2024. doi:10.1007/S00453-024-01261-0.

11 Martin Grohe and Stephan Kreutzer. Methods for algorithmic meta theorems. *AMS-ASL Joint Special Session*, 558:181–206, 2009.

12 Jan van den Heuvel. The complexity of change. In Simon R. Blackburn, Stefanie Gerke, and Mark Wildon, editors, *Surveys in Combinatorics 2013*, volume 409 of *London Mathematical Society Lecture Note Series*, pages 127–160. Cambridge University Press, 2013. doi:10.1017/CBO9781139506748.005.

13 Michael Lampis and Valia Mitsou. Fine-grained meta-theorems for vertex integrity. *Log. Methods Comput. Sci.*, 20(4), 2024. doi:10.46298/LMCS-20(4:18)2024.

14 Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, and Marcin Wrochna. Reconfiguration over tree decompositions. In *IPEC 2014*, volume 8894 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2014. doi:10.1007/978-3-319-13524-3_21.

15 Naomi Nishimura. Introduction to reconfiguration. *Algorithms*, 11(4):52, 2018. doi:10.3390/a11040052.

16 Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, 1994.

17 Marcin Wrochna. Reconfiguration in bounded bandwidth and tree-depth. *J. Comput. Syst. Sci.*, 93:1–10, 2018. doi:10.1016/j.jcss.2017.11.003.