Fragments of Fixpoint Logic on Data Words*

Thomas Colcombet[1] and Amaldev Manuel[2]

1 LIAFA, CNRS, Université Paris 7-Paris Diderot
thomas.colcombet@liafa.univ-paris-diderot.fr

2 MIMUW, University of Warsaw
amal@mimuw.edu.pl

Abstract

We study fragments of a μ-calculus over data words whose primary modalities are '*go to next position*' ($\mathtt{X}^g$), '*go to previous position*' ($\mathtt{Y}^g$), '*go to next position with the same data value*' ($\mathtt{X}^c$), '*go to previous position with the same data value* ($\mathtt{Y}^c$)'. Our focus is on two fragments that are called the bounded mode alternation fragment (BMA) and the bounded reversal fragment (BR). BMA is the fragment of those formulas that whose unfoldings contain only a bounded number of alternations between global modalities ($\mathtt{X}^g, \mathtt{Y}^g$) and class modalities ($\mathtt{X}^c, \mathtt{Y}^c$). Similarly BR is the fragment of formulas whose unfoldings contain only a bounded number of alternations between left modalities ($\mathtt{Y}^g, \mathtt{Y}^c$) and right modalities ($\mathtt{X}^g, \mathtt{X}^c$). We show that these fragments are decidable (by inclusion in Data Automata), enjoy effective Boolean closure, and contain previously defined logics such as the two variable fragment of first- order logic and DataLTL. More precisely the definable language in each formalism obey the following inclusions that are effective.

$$\mathrm{FO}^2 \subsetneq \mathrm{DataLTL} \subsetneq \mathrm{BMA} \subsetneq \mathrm{BR} \subsetneq \nu \subseteq \text{Data Automata}\ .$$

Our main contribution is a method to prove inexpressibility results on the fragment BMA by reducing them to inexpressibility results for combinatorial expressions. More precisely we prove the following hierarchy of definable languages,

$$\emptyset = \mathrm{BMA}^0 \subsetneq \mathrm{BMA}^1 \subsetneq \cdots \subsetneq \mathrm{BMA} \subsetneq \mathrm{BR}\ ,$$

where BMA^k is the set of all formulas whose unfoldings contain at most $k-1$ alternations between global modalities ($\mathtt{X}^g, \mathtt{Y}^g$) and class modalities ($\mathtt{X}^c, \mathtt{Y}^c$). Since the class BMA is a generalization of FO^2 and DataLTL the inexpressibility results carry over to them as well.

1998 ACM Subject Classification F.4.1

Keywords and phrases Data words, Data automata, Fixpoint logic

1 Introduction

Data words are words of the form $(a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times \mathcal{D})^*$ where Σ is a finite set of letters and $\mathcal{D}$ is an infinite domain of *data values*. Typically the alphabet Σ abstracts a finite set of actions or events and the set of data values $\mathcal{D}$ models some sort of identity information. Thus, data words can model a number of scenarios where the information is linearly ordered and it is composed of finite as well as unbounded elements. For example the authors of [1] imagine Σ as the actions of a finite program and $\mathcal{D}$ as process ids. Then, an execution trace of a system with unbounded instances of the program can be modeled as

* The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 259454.

a data word in which each action is associated with the identifier of the process which has generated it.

The paradigmatic question in the study of data words is to develop suitable models (in particular automata and logics) to specify properties of data words. Sure enough there exists a rich variety of models for specifying properties of data words that includes Data Automata [4], Register Automata [15, 9], Pebble Automata [19], Class Memory Automata [1], Class Automata [2], Walking Automata [18], Variable Automata [13], First-Order logic with two variables [4], guarded MSO logic [5], DataLTL [16], Freeze-Logics[9, 14], Logic of Repeating Values [8], XPath [11, 12], Regular expressions [17], Data Monoids [3] etc.

In this work we further study a modal fixpoint logic on data words that we introduced in [6]. This logic is composed of four modalities that allow to evaluate formulas on the successor, class successor (the nearest future position with the same data value), predecessor and class predecessor (nearest past position with the same data value) positions, $\mathtt{Y}^g, \mathtt{Y}^c$. In addition there is a couple of zeroary modalities that describes whether these positions coincide or not. To build the formulas, besides the usual Boolean operations, it is allowed to form the least and greatest fixpoints of formulas . In [6] it is shown that the satisfiability problem for the set of formulas that use only least fixpoints is undecidable, whereas the fragment that consists of only greatest fixpoints is subsumed by Data Automata and hence it has a decidable satisfiability problem. The main result of the work was the decidability of an alternation-free fragment of the logic that further bounds the number of change of directions in evaluating the formulas by using a generalisation of Data Automata.

Contributions

In the present paper, we aim at restricting the power of the above μ-calculus logic for data words for obtaining classes that are closed under all Boolean connectives, mirroring, and enjoy decidability of emptiness and universality. We consider two restricted fragments that achieve this goal. The first one, called BMA (for Bounded Mode Alternation) syntactically bounds the number of changes between class and global modes. The second, called BR (for Bounded Reversal), syntactically bounds the number of changes between left modalities and right modalities.

It is easy to show that BMA is contained in Data Automata. It is not very difficult to show that BMA is subsumed by BR, that is to say for every formula in BMA there is an equivalent one in BR. Our main result is the strictness of this last inclusion, i.e. that there is a formula in BR for which there is no equivalent formula in BMA. This proof uses a deep result from combinatorics called the Hales-Jewett theorem. As a proof device we use a sort of circuits called *combinatorial expressions* that were introduced in [7]. These expressions define functions over an infinite domain (for instance the integers). They are built by composing *gates* that are functions of two kinds, either the function has a bounded arity, or the function has a bounded domain. In [7] it is shown that certain properties (a property is a function that has a binary codomain) for instance *the given sequence of positive integers has gcd 1* or *the given sequence of integers sum to 0* cannot be computed by expressions of fixed depth. We use a variant of this theorem in this paper to show that there is a formula in BR for which there is no equivalent one in BMA. More precisely it is shown that there is a specific formula in BR such that if it has an equivalent formula in BMA, then it is possible to construct expressions of fixed depth for a particular property and since that particular property cannot be computed by fixed depth expressions, we derive a contradiction. One thing to note is that since the techniques developed in [7] are general enough to derive impossibility results for a large family of properties, correspondingly the proof method developed here can be used to show inexpressibility results for a variety of formulas.

Now we examine the implications of our result in a larger context. As mentioned earlier, the results mentioned here have very close connection with Data Automata (DA for short). The well known feature of DA is that it subsumes the logic $\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c 1)$ on data words where Σ denotes the unary predicates indicating the letters, $<$ is the linear order on positions, $+1$ is the successor relation on positions, $\sim$ is the equivalence relation on positions with respect to the data values (i.e. $i \sim j$ if $d_i = d_j$), and $+^c 1$ is the class successor relation. It is known that data languages recognisable by DA are closed under union, intersection and letter-to-letter projection, but not under complementation [4]. Since FO^2 formulas are closed under Boolean operations, it is evident that Data Automata strictly subsumes the logic FO^2. This observation prompts the question that if there are other classes subsumed by DA that are closed under Boolean operations. The fragments introduced in the paper answer this question positively. Note only that, there are automata theoretic characterisations that are natural variants of DA for both these fragments (we only present the one for BMA).

Another and perhaps more important question is how to show that a given data language is not expressible in FO^2. Note that in some cases, using the techniques on words over finite alphabets it is possible to show that a given data language is not definable in FO^2 (for instance to show that data words of even length are not definable in FO^2). We are interested in those cases where such reductions are not possible, in particular where the property given is dependent on the data values. We don't have a complete solution to this problem yet, but our method to prove inexpressibility results on BMA offers a partial answer. This is because the logic $\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c 1)$, as it is shown in this paper, is equivalent to the unary fragment of a temporal logic, namely DataLTL [16], which is a strict subfragment of BMA. DataLTL is the temporal logic where usual temporal operators such as *until, future, past* etc. exist both on the linear order on positions (called the global order) as well as on the suborders formed by subsets of positions that share the same data value (called the class orders). For instance the temporal operator $\mathtt{F}^g\varphi$ is true a position if there is a position in the future that satisfies the formula φ, whereas the formula $\mathtt{F}^c\varphi$ is true at a position if there is a future position that has the same data value as the current position and that satisfies the formula φ. The unary fragment of DataLTL is the subclass of formulas that uses only the unary temporal operators (such as $\mathtt{F}^g, \mathtt{P}^g, \mathtt{F}^c$ etc). Since every such operator is expressible in FO^2 it is immediate that unary DataLTL is subsumed by the logic FO^2. But the converse direction, which is shown in the paper, is not obvious, since it is not immediate how to translate formulas like $\exists y\,(a(x) \wedge b(y) \wedge x < y \wedge x \not\sim y)$. Thus inexpressibility results on the fragment BMA renders directly corresponding results on all sublogics including FO^2 and DataLTL.

Finally let us also note that the translations outlined in this paper, namely

$$\mathrm{FO}^2 \subsetneq \mathrm{DataLTL} \subsetneq \mathrm{BMA} \subsetneq \mathrm{BR} \subsetneq \nu \subseteq \mathrm{DA},$$

constitutes an alternate proof the main result of [4] that FO^2 is subsumed by DA. The proof in [4] is a direct translation of FO^2 formulas by a intricate case analysis. Our proof, however, is modular and makes use of analogous constructions from automata theory on finite words.

Related work

As mentioned already this work is strongly related to DataLTL, FO^2 and DA. One other very popular ecosystem on data words is that of Register Automata and the associated lo-

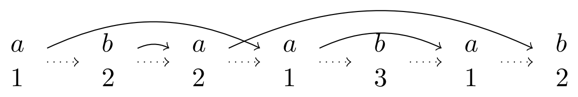

Figure 1 A data word and its graph. Dotted and thick arrows denote the successor and class successor relations respectively.

gics such as Freeze LTL, Freeze μ-calculus, Xpath [9, 14, 8, 11, 12] etc. Our inexpressibility result implies that BMA is incomparable to Register Automata (in particular nondeterministic 1-Register Automata). Since all our modalities are expressible in terms of successor, predecessor, freeze operator and fixpoint operators, our fixpoint logic is subsumed by Freeze μ-calculus of [14]. However it should be noted that the latter logic is highly undecidable [9]. The decidable fragment of Freeze μ-calculus (and also Freeze LTL) is unidirectional (only future modalities) but our logic is naturally two-way. Finally the decidable two-way fragment of Freeze LTL, namely Simple Freeze LTL is equivalent to FO^2 and hence it is subsumed by BMA. Therefore our method of proving inexpressibility extends to this logic as well.

Structure of the document

In Section 2 we present the definition of our fixpoint logic and give some examples. In Section 3 we recall the *composition* operator (*comp*) on sets of formulas and define the fragments BMA and BR using it. Thereafter, a characterisation of the class BMA in terms of cascades of automata, that is used in the proof of the separation theorem, is given. In Section 4, first we recollect the paradigm of combinatorial expressions and state the necessary results for our purpose. Afterwards it shown how to translate a cascade on data words with a specific structure to expressions and the separation theorem is proved. In Section 5 we conclude. Due to space constraints some proofs have been moved to the appendix.

2 μ-Calculus on Data Words

In this section, we recall the basics of the μ-calculus on data words [6].

Fix an infinite set $\mathcal{D}$ of *data values*. *Data words* are words of the form $u = (a_1, d_1) \cdots (a_n, d_n) \in (\Sigma \times \mathcal{D})^*$ where Σ is a finite set of *letters*. A maximal set of positions in u with the same data value is called a *class*. The set of classes in u define an equivalence relation $\sim$, called the *class relation*, on the set of positions of u. Given a permutation σ of $\mathcal{D}$, it can be applied on a data word as expected, yielding $\sigma(u) = (a_1, \sigma(d_1)) \dots (a_n, \sigma(d_n))$. The data words u and $\sigma(u)$ have the same class relation. A *data language* is a set of data words that is invariant under such applications of the permutations of $\mathcal{D}$.

For our purposes, it is convenient to see data words as graphs in the following manner. To each data word $w = (a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times \mathcal{D})^*$ associate the graph $G_w = ([n], \ell, +1, +^c1)$ where $[n]$ is the set of positions $\{1, \dots, n\}$, $\ell : \Sigma \to 2^{[n]}$ is the labelling function $\ell(a) = \{i \mid a_i = a\}$, the binary relation $+1$ denotes the *successor* relation on positions, *i.e.*, $+1(i, j)$ if $j = i + 1$, and the binary relation $+^c1$ denotes the *class successor* relation on positions, *i.e.*, $+^c1(i, j)$ if $i < j$, $d_i = d_j$, and $d_m \neq d_i$ for all $i < m < j$. We call *predecessor* relation (*resp.*, *class predecessor* relation) the reverse of the successor relation (*resp.*, class successor relation). We implicitly identify a data word with its graph. Figure 1 shows a data word and its corresponding graph.

Seen as such graphs, data words are naturally prone to the use of temporal logics. Let $\mathit{Prop} = \{p, q, \ldots\}$ and $\mathit{Var} = \{x, y, \ldots\}$ be countable sets of *propositional variables* and *fixpoint variables* respectively. The *μ-calculus* on data words is the set of all *formulas* φ respecting the following syntax:

$$\varphi := x \mid A \mid \neg A \mid \mathtt{M}\,\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu x.\varphi \mid \nu x.\varphi$$
$$\text{where} \quad \mathtt{M} := \mathtt{X}^g \mid \mathtt{X}^c \mid \mathtt{Y}^g \mid \mathtt{Y}^c \qquad \text{and} \quad A := p \in \mathit{Prop} \mid \mathcal{S} \mid \mathcal{P} \mid \mathsf{fst}^c \mid \mathsf{fst}^g \mid \mathsf{lst}^c \mid \mathsf{lst}^g$$

The elements of $\mathtt{M}$ are called *modalities*, and the ones of A, atoms. The set of zeroary modalities $\{\mathsf{fst}^c, \mathsf{fst}^g, \mathsf{lst}^c, \mathsf{lst}^g, \mathcal{P}, \mathcal{S}\}$ will be denoted by the symbol Z for the rest of the paper.

The semantic of a formula φ, over a data word w is the set of positions of w where "φ is true" (denoted as $[\![\varphi]\!]_w$). The formal definition is given in Figure 2. The different constructs have their expected meaning, keeping in mind that the class modalities $\mathtt{X}^c, \mathtt{Y}^c, \mathsf{fst}^c, \mathsf{lst}^c$ have to be interpreted on the word restricted to the current data value. The modality $\mathcal{S}$ (*resp.*, $\mathcal{P}$) holds at a position i if the successor and class successor i coincide (*resp.* the predecessor and class predecessor coincide).

$$\begin{aligned}
[\![\mathsf{fst}^g]\!]_w &= \{1\} & [\![\mathtt{X}^g\varphi]\!]_w &= [\![\varphi]\!]_w - 1\\
[\![\mathsf{lst}^g]\!]_w &= \{n\} & [\![\mathtt{Y}^g\varphi]\!]_w &= [\![\varphi]\!]_w + 1\\
[\![\mathsf{fst}^c]\!]_w &= \{i \mid \nexists j = i -^c 1\} & [\![\mathtt{X}^c\varphi]\!]_w &= [\![\varphi]\!]_w -^c 1\\
[\![\mathsf{lst}^c]\!]_w &= \{i \mid \nexists j = i +^c 1\} & [\![\mathtt{Y}^c\varphi]\!]_w &= [\![\varphi]\!]_w +^c 1\\
[\![\varphi_1 \wedge \varphi_2]\!]_w &= [\![\varphi_1]\!]_w \cap [\![\varphi_2]\!]_w & [\![\mathcal{S}]\!]_w &= \{i \mid i+1 = i +^c 1\}\\
[\![\varphi_1 \vee \varphi_2]\!]_w &= [\![\varphi_1]\!]_w \cup [\![\varphi_2]\!]_w & [\![\mathcal{P}]\!]_w &= \{i \mid i-1 = i -^c 1\}\\
[\![\mu x.\varphi]\!]_w &= \cap\{S \subseteq [n] \mid [\![\varphi]\!]_{w[\ell(x):=S]} \subseteq S\} & [\![p]\!]_w &= \ell(p)\\
[\![\nu x.\varphi]\!]_w &= \cup\{S \subseteq [n] \mid S \subseteq [\![\varphi]\!]_{w[\ell(x):=S]}\} & [\![\neg p]\!]_w &= [n] \setminus \ell(p)\\
[\![x]\!]_w &= \ell(x)
\end{aligned}$$

Figure 2 Semantics of μ-calculus on data words $w = ([n], +1, +^c 1, \ell)$.

Note that in this definition of the logic, negations in a formula are located at the leaves. It is nevertheless possible, as usual, to negate such formulas by pushing the negation toward the leaves, but this requires a bit of care when negating modalities. For instance, $\neg\mathtt{X}^c\varphi$ is not equivalent to $\mathtt{X}^c\neg\varphi$, but to $\mathsf{lst}^c \vee \mathtt{X}^c\neg\varphi$. Similar arguments have to be used for all modalities. Following these ideas, we define the dual modalities $\tilde{\mathtt{X}}^g\varphi \equiv \mathsf{lst}^g \vee \mathtt{X}^g\varphi$, $\tilde{\mathtt{Y}}^g\varphi \equiv \mathsf{fst}^g \vee \mathtt{Y}^g\varphi$, $\tilde{\mathtt{X}}^c\varphi \equiv \mathsf{lst}^c \vee \mathtt{X}^c\varphi$ and $\tilde{\mathtt{Y}}^c\varphi \equiv \mathsf{fst}^c \vee \mathtt{Y}^c\varphi$. These modalities are considered dual since $\tilde{\mathtt{X}}^g\varphi \equiv \neg\mathtt{X}^g\neg\varphi$, ...

Next we lay out some terminology and abbreviations which we will use in the subsequent sections. Let λ denote either μ or ν. Every occurrence of a fixpoint variable x in a subformula $\lambda x.\psi$ of a formula is called *bound*. All other occurrences of x are called *free*. A formula is called a *sentence* if all the fixpoint variables in φ are bound. If $\varphi(x_1, \ldots, x_n)$ is a formula with free variables $x_1, \ldots, x_n$, then by $\varphi(\psi_1, \ldots, \psi_n)$ we mean the formula obtained by substituting ψ_i for each x_i in φ. As usual the bound variables of $\varphi(x_1, \ldots, x_n)$ may require a renaming to avoid the capture of the free variables of ψ_i's. For a sentence φ and a position i in the word w, we denote by $w, i \models \varphi$ if $i \in [\![\varphi]\!]_w$. The notation $w \models \varphi$ abbreviates the case when $i = 1$. The *data language* of a sentence φ, denoted as $L(\varphi)$, is the set of data words w such that $w \models \varphi$.

By *μ-fragment* we mean the subset of μ-calculus which uses only μ-fixpoints. Similarly *ν-fragment* stands for the subset which uses only ν-fixpoints.

▶ **Example 1** (temporal modalities)**.** An example of a formula would be $\varphi\,\mathtt{U}^g\,\psi$ which holds if ψ holds in the future, and φ holds in between. This can be implemented as $\mu x.\psi \vee (\varphi \wedge \mathtt{X}^g x)$ The formula $\varphi \mathtt{U}^c \psi = \mu x.\psi \vee (\varphi \wedge \mathtt{X}^c x)$ is similar, but for the fact that it refers only to the class of the current position. The formula $\mathtt{F}^g\varphi$ abbreviates $\top\,\mathtt{U}^g\,\varphi$, and its dual is $\mathtt{G}^g\varphi = \neg\mathtt{F}^g\neg\varphi$. The constructs $\mathtt{S}^g$, $\mathtt{S}^c$, $\mathtt{P}^g$, $\mathtt{P}^c$, $\mathtt{H}^g$ and $\mathtt{H}^c$, are defined analogously, using past modalities, and correspond respectively to $\mathtt{U}^g$, $\mathtt{U}^c$, $\mathtt{F}^g$, $\mathtt{F}^c$, $\mathtt{G}^g$ and $\mathtt{G}^c$. For instance, $\mathtt{F}^c\mathtt{P}^c\varphi$ expresses that there is a position in the class that satisfies φ and $\mathtt{F}^c\mathtt{P}^c(\varphi \wedge \tilde{\mathtt{X}}^c\mathtt{G}^c\neg\varphi \wedge \tilde{\mathtt{Y}}^c\mathtt{H}^c\neg\varphi)$ expresses that there exists exactly one position which satisfies φ in the class.

3 The bounded reversal and bounded mode alternation fragments

In this section we introduce the bounded mode alternation and bounded reversal fragments (BMA and BR) and compare these two fragments between themselves and with other logics (Theorem 5).

Before delving into the technical details let us outline the intuition behind each of the fragments. The four modalities $\mathtt{X}^g, \mathtt{Y}^g, \mathtt{X}^c$ and $\mathtt{Y}^c$ can be divided along two axis. Based on the directions: there are the *left modalities* $\mathtt{Y}^g$, $\mathtt{Y}^c$, and *right modalities* $\mathtt{X}^g$, $\mathtt{X}^c$. Based on the modes: there are *global modalities* $\mathtt{X}^g$, $\mathtt{Y}^g$, and *class modalities* $\mathtt{X}^c$, $\mathtt{Y}^c$. The BR and BMA fragments are defined by limiting the number of alternation between this types of modalities. This is formally achieved using the operation *comp* that we define now.

Let Ψ be a set of μ-calculus formulas. Define the sets $comp^i(\Psi)$ for $i \in \mathbb{N}$ inductively

- $comp^0(\Psi) = \emptyset$,
- $comp^{i+1}(\Psi) = \{\psi(\varphi_1, \ldots, \varphi_n) \mid \psi(x_1, \ldots, x_n) \in \Psi,\ \varphi_1, \ldots, \varphi_n \in comp^i(\Psi)\}$ in which the substitution $\psi(\varphi_1, \ldots, \varphi_n)$ is allowed only if none of the free variables of $\varphi_1, \ldots, \varphi_n$ get bound in $\psi(\varphi_1, \ldots, \varphi_n)$.

The set of formulas $comp(\Psi)$ is defined as $comp(\Psi) = \bigcup_{i\in\mathbb{N}} comp^i(\Psi)$. For a formula $\psi \in comp(\Psi)$, the *comp-height* of ψ in $comp(\Psi)$ in the least i such that ψ is in $comp^i(\Psi)$.

We are now ready to define the BR and BMA fragments of the μ-calculus. For a set of modalities M, define *formulas*(M) to be the set of formulas that uses only the modalities M apart from the zeroary modalities.

▶ **Definition 2.** The BMA and the BR fragments of μ-calculus are respectively:

$$\mathrm{BMA} = comp\left(formulas\left(\{\mathtt{X}^g, \mathtt{Y}^g\}\right) \cup formulas\left(\{\mathtt{X}^c, \mathtt{Y}^c\}\right)\right),$$

and

$$\mathrm{BR} = comp\left(formulas\left(\{\mathtt{X}^g, \mathtt{X}^c\}\right) \cup formulas\left(\{\mathtt{Y}^g, \mathtt{Y}^c\}\right)\right).$$

Further, BMA^k denotes the subset of BMA with comp-height k. Similarly BR^k stands for the subset of BR with comp-height k.

▶ **Example 3.** Let

$$\varphi_1 = \nu x.(\mathtt{X}^c x \vee \mathtt{X}^g \mu y.(q \wedge \mathtt{Y}^c y)), \qquad \varphi_2 = \nu x.\left(\mathtt{X}^c\mathsf{lst}^g \vee \mathtt{X}^c\mathtt{Y}^g x\right),$$

$$\varphi_3 = \mu x.((\nu y.\, q \vee \mathtt{X}^c y) \vee \mathtt{X}^g x \vee \mathtt{Y}^g x), \quad \text{and} \quad \varphi_4 = \mu x.(\mathtt{X}^c\mathtt{X}^g x \vee p).$$

The formula φ_1 is in BR^2 and in BMA^3. The formula φ_2 is neither in BR nor in BMA. The formula φ_3 is in BMA^2 but not in BR. The formula φ_4 is in BR^1 but not in BMA.

▶ **Example 4.** Consider the language L_k that contains the data words such that, by applying k-times the sequence of the global successor followed by the class successor, one reaches a position labeled with letter a. The language L is the union of all L_k for k ranging over all non-negative integers. The language L_k is defined by φ_k and L by φ defined as follows:

$$\varphi_k = \overbrace{\mathtt{X}^g\mathtt{X}^c \ldots \mathtt{X}^g\mathtt{X}^c}^{k\text{-times}} a, \qquad \text{and} \qquad \varphi = \mu x.(\mathtt{X}^g\mathtt{X}^c x \vee a) \ .$$

The formula φ_k is in BR^1 and in BMA^{2k}. The formula φ is in BR^1, but not in BMA. Later in Section 4 we will prove that a variant of L is not definable by any formula in BMA.

Let us now state the main theorem of this section, namely the inclusions between the fragments of the μ-calculus in terms of the data languages defined. Below *DataLTL* denotes the temporal logic on data words consisting of the modalities $\{\mathcal{S}, \mathcal{P}, \mathtt{X}^g, \mathtt{Y}^g, \mathtt{X}^c, \mathtt{Y}^c, \mathtt{U}^g, \mathtt{S}^g, \mathtt{U}^c, \mathtt{S}^c\}$, *uDataLTL* is the unary sublogic consisting of the modalities $\{\mathcal{S}, \mathcal{P}, \mathtt{X}^g, \mathtt{X}^c, \mathtt{Y}^g, \mathtt{Y}^c, \mathtt{F}^c, \mathtt{F}^g, \mathtt{P}^g, \mathtt{P}^c\}$ and ν denotes the fragment of the μ-calculus containing only the greatest fixpoints (ν's).

▶ **Theorem 5.** *The following inclusions hold for definable languages,*

$$\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c1) = \mathit{uDataLTL} \subsetneq \mathit{DataLTL} \subsetneq \mathrm{BMA} \subseteq \mathrm{BR} \subsetneq \nu \subseteq \mathrm{DA} \ .$$

3.1 Characterizing BMA as cascades of automata

Next we give a characterization of BMA in terms of cascades of finite state automata. It is classical that composition (*comp*) corresponds to the natural operation of composing sequential transducers that compute subset of subformulas that are true at each position. Given a μ-calculus formula φ over words, we can see it as a transducer that reads the input, and labels every position with one extra bit of information denoting the truth value of the formula φ at that position. Under this view, the composition of formulas corresponds to applying the transducers in sequence: the first transducer reads the input, and adds some extra labelling on it. Then a second transducer reads the resulting word, and processes it in a similar way, etc... If we push this view further, we can establish exact correspondences between the class BMA, and suitable cascades of transducers. Furthermore, the comp-height of the formula matches the number of transducers involved in the cascade.

First we introduce some notation. Given a data word $w = (a_1, d_1) \cdots (a_n, d_n)$ the *string projection* of w, denoted by $\mathit{str}(w)$, is the word $a_1 \cdots a_n$. For a class $S = \{i_1, \ldots, i_m\}$ the *class projection* corresponding to S, denoted as $\mathit{str}(w|_S)$, is the finite word $a_{i_1} \cdots a_{i_m}$. For a word $u = b_1 \cdots b_n$, the relabelling of w by u is the data word $(b_1, d_1) \ldots (b_n, d_n)$. Similarly the relabelling of the class S in w by $b_1 \cdots b_m$ is the data word $(a'_1, d_1) \cdots (a'_n, d_n)$ where $a'_i = b_j$ if $i = i_j$ and a_i otherwise.

The *marking* of a position i in the data word w, in notation $m(i)$, is the subset of zeroary modalities Z satisfied by i. The *marked string projection* of w, denoted as $\mathrm{mstr}(w)$, is the word $(a_1, m(i)) \cdots (a_n, m(n))$ over the alphabet $\Sigma \times 2^Z$. Given a class $S = \{i_1, \ldots, i_n\}$ the *marked class projection* of S, denoted as $\mathrm{mstr}(w|_S)$, is the finite word $(a_{i_1}, m(i_1)) \cdots (a_{i_n}, m(i_n))$.

A *functional letter-to-letter transducer* $\mathcal{A} : \Sigma^* \to \Gamma^*$ over words is a nondeterministic finite state letter-to-letter transducer such that every input word $w \in \Sigma^*$ has at most one output word $\mathcal{A}(w) \in \Gamma^*$.

We next disclose two forms of transductions possible by a word transducer on data words. Let $\mathcal{A} : (\Sigma \times 2^Z)^* \to \Gamma^*$ be a functional letter-to-letter transducer. The automaton $\mathcal{A}$ acts as a *global transducer* when it runs on the marked string projection $\mathrm{mstr}(w)$ of the input data word $w \in (\Sigma \times \mathcal{D})^*$. If the run succeeds then the unique output data word $w' \in (\Gamma \times \mathcal{D})^* = \mathcal{A}(w)$ (by abuse of notation) is the relabelling of w with the word $\mathcal{A}(\mathrm{mstr}(w))$. Automaton $\mathcal{A}$ is a *class transducer* when for each class S in the input data word w, a copy of the automaton $\mathcal{A}$ runs on the marked class projection $\mathrm{mstr}(w|_S)$. If all the runs succeed then the unique output data word $\mathcal{A}(w)$ (by abuse of notation) is the relabelling of each class of S of w by $\mathrm{mstr}(w|_S)$.

▶ **Definition 6.** A *cascade of class and global transducers over data words* (hereafter simply cascade) $\mathcal{C}$ is a sequence $\langle \Sigma = \Sigma_0, \mathcal{A}_1, \Sigma_1, \dots, \Sigma_{n-1}, \mathcal{A}_n, \Sigma_n \rangle$ such that $\mathcal{A}_1, \dots, \mathcal{A}_n$ is a sequence of class and global transducers over data words and for each i, the transducer $\mathcal{A}_i$ has input alphabet $\Sigma_{i-1} \times 2^Z$ and output alphabet Σ_i. Sets Σ_0, Σ_n are respectively the *input* and *output* alphabets of the cascade $\mathcal{C}$ and n is the *height* of the cascade.

The cascade $\mathcal{C}$ has a *successful run* on a given data word w if there is a sequence of data words $w_0 = w, w_1, \dots, w_{n-1}, w_n$ such that each transducer $\mathcal{A}_i$ has a successful run on w_{i-1} outputing the data word w_i. The data word w_n is the output of the cascade $\mathcal{C}$, in notation $\mathcal{C}(w) = w_n$. The language accepted by the cascade $\mathcal{C}$, denoted as $L(\mathcal{C})$, is the set of all data words w on which $\mathcal{C}$ has a successful run.

Two cascades $\mathcal{C}_1$ and $\mathcal{C}_2$ can be composed to form the cascade $\mathcal{C}_1 \circ \mathcal{C}_2$ if the output alphabet of $\mathcal{C}_1$ and the input alphabet of $\mathcal{C}_2$ are the same. Composition of cascades is the natural analogue of composition of formulas, this is expressed by the following proposition.

▶ **Proposition 7.** *Let L be a set of data words. Then the following statements are equivalent.*

1. *L is definable by a formula in* BMA *of comp-height k.*
2. *L is recognisable by a cascade of height k.*

4 Seperation of the fragments BMA and BR

In this section we prove the main theorem of the paper, namely the separation of the fragments of BMA and BR. More precisely it is shown that there is a formula in BR that has no equivalent formula in BMA. We start by presenting our technical tool, namely combinatorial expressions [7].

4.1 Combinatorial expressions

Put simply, combinatorial expressions are *circuits* over a *data domain* $\mathcal{E}$. For our purposes it is sufficient to assume that $\mathcal{E}$ is a set that contains all the usual data types such as Booleans, integers, finite words etc. We form expressions by composing variables (denoted by X, Y etc.) and functions (denoted by f, g etc.) whose domains and ranges are explicitly specified. A variable X has *range* $E \subseteq \mathcal{E}$, denoted as $X : E$, if it takes values from the set E. We say a function $f : E_1 \times \dots \times E_k \to F$, where $E_1, \dots, E_k, F \subseteq \mathcal{E}$, has *arity* k, *domain* $E_1 \times \dots \times E_k$ and *range* F. The expressions are built using two specific classes of functions (called gates), namely:

- *binary functions* — when $k \leq 2$, and,
- *finitary functions* — when each of $E_1, \dots, E_k$ is finite.

For example the addition on integers $+ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ is a binary function, whereas the Boolean disjunction of k inputs $\vee : \{0,1\}^k \to \{0,1\}$ is a finitary function.

▶ **Definition 8.** *Combinatorial expressions* are defined inductively;

- a variable $X : E$ is a combinatorial expression with *range* E, and *depth* 0.

- if $f : E_1 \times \cdots \times E_k \to F$ is a binary or a finitary function, and $t_1, \ldots, t_k$ are combinatorial expressions with ranges $E_1, \ldots, E_k$ and depths $d_1, \ldots, d_k$ respectively, then $f(t_1, \ldots, t_k)$ is a combinatorial expression with range F and depth $\max(d_1, \ldots, d_k) + 1$.

Let $t(\bar{X})$ be a combinatorial expression that contains (possibly vacuously) the variables $\bar{X} = X_1 : E_1, \ldots, X_n : E_n$. For the valuation $\bar{a} = a_1, \ldots, a_n$, where $a_i \in E_i$ for each i, of the variables $\bar{X}$, the value of the expression t, denoted as $t(\bar{a})$, is defined inductively; if t is a variable X_i then $t(\bar{a}) = a_i$, and if $t = f(t_1, \ldots, t_k)$ then $t(\bar{a}) = f(t_1(\bar{a}), \ldots, t_k(\bar{a}))$. Assume $F \subseteq \mathcal{E}$ is the range of the expression t. Naturally t defines a map $[\![t]\!] : \bar{a} \to t(\bar{a})$ from the set $E_1 \times \cdots \times E_n$ to the set F. Given a map $m : E_1 \times \cdots \times E_n \to F$, where $E_1, \ldots, E_n, F \subseteq \mathcal{E}$, we say the map is *recognised* by an expression t if $[\![t]\!] = m$. A particular case is when the range of the map m is restricted to a set of size two (without loss of generality $\{0, 1\}$); in which case we say that t *recognises* the *property* $\{a_1, \ldots, a_n : m(a_1, \ldots, a_n) = 1\}$.

▶ **Example 9.** Each map $f : E^n \to F$, for some $E, F \subseteq \mathcal{E}, n \in \mathbb{N}$, has an expression of depth $\lceil \log n \rceil + 1$ recognising it. Let $cat : E^* \times E^* \to E^*$ be the concatenation operation on words over the alphabet E and let $t(X_1 : E, \ldots, X_n : E)$ be an expression of depth $\lceil \log n \rceil$ that consists of only the function cat and that computes the concatenation of the inputs. Let $u : E^* \to F$ be a binary function on words over E such that $u(e_1 \cdots e_n) = f(e_1, \ldots, e_n)$. The map f is recognised by expression $u(t(X_1 : E, \ldots, X_n : E))$.

▶ **Example 10.** Consider the set P_n of n-tuples $(u_1, \ldots, u_n)$ of words in $\{0, 1\}^*$ that all have equal length. The property P_n is recognised by the expression

$$t = \bigwedge \left(el\left(X_1, X_2\right), \ldots, el\left(X_1, X_n\right), el\left(X_2, X_3\right), \ldots, el\left(X_2, X_n\right), \ldots, el(X_{n-1}, X_n) \right)$$

where $\bigwedge$ is the Boolean conjunction on $n \cdot (n-1)/2$ inputs and $el : A^* \times A^* \to \{0, 1\}$ is the function on words defined as $el(u, v) = 1$ iff the words u and v are of the same length. The function $\bigwedge$ is finitary and the function el is binary. The expression t has depth 2.

In the previous example, regardless of the value of n the expression t has a constant depth. But there exists properties for which it is not the case.

▶ **Definition 11.** Let V_n be the set of n-tuples $(u_1, \ldots, u_n)$ of words over the alphabet $\{0, 1\}$ such that:

1. the words $u_1, \ldots, u_n$ are of the same length, and;
2. there exists a position $1 \leq i \leq |u_1|$ such that the ith letter of each of u_1 to u_n is 1.

It is shown in [7] that,

▶ **Theorem 12.** *There is no expression of depth at most k that recognises the property V_{2^k+1}.*

4.2 Application

We now apply the above theorem to derive our inexpressibility results. The idea is to define a data language B_n that corresponds to the property V_n and to show that if there is a BMA-formula of comp-height k recognising B_n then there is a combinatorial expression of depth $\mathcal{O}(k)$ (precise bound disclosed later) recognising the property V_n. This claim along with the Theorem 12 implies a lower bound on the comp-height of formulas defining the language B_n.

For the proof we rely on data words with a special structure that encode a sequence of words. Let $v_1, \ldots, v_n \in \Sigma^*$ be words of identical and even length, say $2\ell \in \mathbb{N}$. A data word

$w \in (\Sigma \times \mathcal{D})^*$ is a *coding* of the words $v_1, \ldots, v_n \in \Sigma^*$, denoted as $w = \mathit{coding}(v_1, \ldots, v_n)$, if $w = w_1 \cdots w_n$ with $v_1 = \mathit{str}(w_1), \ldots, v_n = \mathit{str}(w_n)$ and the class relation is the set of tuples $(k \cdot 2\ell + 2i, (k+1) \cdot 2\ell + 2i - 1)$ for $0 \le k < n-1, 1 \le i \le \ell$; the position $k \cdot 2\ell + 2i$ is the ith even position in the block w_{k+1} and $(k+1) \cdot 2\ell + 2i - 1$ is the ith odd position in the block w_{k+2}. Coding is only defined for words of identical even length and hereafter whenever we say $\mathit{coding}(v_1, \ldots, v_n)$ it is understood that $v_1, \ldots, v_n$ are of identical even length.

A data word w is a n-coding (or simply a coding when the value n is clear from the context) if it is the coding of some words $v_1, \ldots, v_n \in \Sigma^*$. We write n-*Codings* for the set of all n-codings.

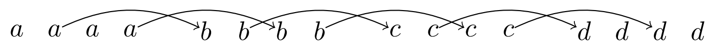

Figure 3 The coding of the words $aaaa, bbbb, cccc, dddd \in \{a, b, c, d\}^*$.

Next we introduce some gates and expressions that we use in the proofs. If w is the coding of $u_1, \ldots, u_n \in \Sigma^*$ then $\mathrm{mstr}(w) = m_1(u_1) \cdot m_2(u_2) \cdots m_2(u_{n-1}) \cdot m_3(u_n)$ for binary gates $m_1, m_2, m_3 : \Sigma^* \to (\Sigma \times 2^Z)^*$ such that: (1) for all words $u = a_1 \cdots a_{2\ell} \in \Sigma^*, 2\ell > 2$

$$m_1(u) = (a_1, x_1) \cdots (a_{2\ell}, x_{2\ell}) \text{ where } x_i = \begin{cases} \{\mathsf{fst}^g, \mathsf{fst}^c, \mathsf{lst}^c\} & \text{if } i = 1, \\ \{\mathsf{fst}^c, \mathsf{lst}^c\} & \text{if } i \text{ is odd and } i \neq 1, \\ \{\mathsf{fst}^c\} & \text{if } i \text{ is even.} \end{cases}$$

$$m_2(u) = (a_1, x_1) \cdots (a_{2\ell}, x_{2\ell}) \text{ where } x_i = \begin{cases} \{\mathsf{lst}^c\} & \text{if } i \text{ is odd,} \\ \{\mathsf{fst}^c\} & \text{if } i \text{ is even.} \end{cases}$$

$$m_3(u) = (a_1, x_1) \cdots (a_{2\ell}, x_{2\ell}) \text{ where } x_i = \begin{cases} \{\mathsf{lst}^c\} & \text{if } i \text{ is odd,} \\ \{\mathsf{fst}^c, \mathsf{lst}^c\} & \text{if } i \text{ is even and } i \neq 2\ell, \\ \{\mathsf{fst}^c, \mathsf{lst}^c, \mathsf{lst}^g\} & \text{if } i = 2\ell. \end{cases}$$

(2) For each word $ab \in \Sigma^2$, $m_1(ab) = (a, \{\mathsf{fst}^c, \mathsf{fst}^g, \mathsf{lst}^c\})(b, \{\mathsf{fst}^c, \mathcal{S}\})$, $m_2(ab) = (a, \{\mathsf{lst}^c, \mathcal{P}\})(b, \{\mathsf{fst}^c, \mathcal{S}\})$, $m_3(ab) = (a, \{\mathsf{lst}^c, \mathcal{P}\})(b, \{\mathsf{fst}^c, \mathsf{lst}^c, \mathsf{lst}^g\})$. (3) For words of odd length the functions m_1, m_2, m_3 are fixed arbitrarily.

Let $\mathit{is\epsilon} : \Sigma^* \to \{0, 1\}$ be the binary gate defined as $\mathit{is\epsilon}(w) = 1$ precisely when $w \in \Sigma^*$ is not the empty word. Let $bI : \Sigma^* \times \{0, 1\} \to \Sigma^*$ be the binary function $bI(x, 1) = x$ and $bI(x, 0) = \epsilon$. For variables $\bar{X} = X_1 : \Sigma^*, \ldots, X_n : \Sigma^*$, let $NE(\bar{X})$ be the expression $\bigwedge(\mathit{is\epsilon}(X_1), \ldots, \mathit{is\epsilon}(X_1))$ of depth 2 that recognises the property that none of the input words is the empty word. Sometimes we use these gates and expressions over other alphabets, and then it is understood that the domains of the functions are appropriately defined.

Next we prove that class transductions and global transductions on n-codings can be defined by expressions of fixed height (irrespective of n). To summarise the intuition, let $w = w_1 \cdots w_n$ be the coding of the words $u_1, \ldots, u_n \in \Sigma^*$ such that $\mathit{str}(w_i) = u_i$. Assume $\mathcal{A} : (\Sigma \times 2^Z)^* \to \Gamma^*$ is a class transducer that has a successful run on w and let $\mathcal{A}(w) = w' = w'_1 \cdots w'_n \in (\Gamma \times \mathcal{D})^*$ where w'_i is a relabelling of w_i. Observe that the only other positions in the class of a position in w_i appear either in w_{i-1} or w_{i+1}. Therefore to compute $\mathit{str}(w'_i)$ it suffices to know the words u_{i-1}, u_i, u_{i+1} and hence there is an expression that takes as inputs u_{i-1}, u_i, u_{i+1} and outputs the word $\mathit{str}(w'_i)$.

▶ **Lemma 13.** *For each class transducer $\mathcal{A} : (\Sigma \times 2^Z)^* \to \Gamma^*$ and each $n \in \mathbb{N}$ there exist combinatorial expressions$e_1(\bar{X}), \ldots, e_n(\bar{X})$, where $\bar{X} = X_1 : \Sigma^*, \ldots, X_n : \Sigma^*$, of depth 7 such that for all n-tuple $\bar{u} = (u_1, \ldots, u_n)$ of words in Σ^* of identical even length $\mathit{coding}(e_1(\bar{u}), \ldots, e_n(\bar{u})) = \mathcal{A}(\mathit{coding}(\bar{u}))$.*

Next we prove a similar claim for global transducers. The idea is as follows. Assume $\mathcal{A} : (\Sigma \times 2^Z)^* \to \Gamma^*$ is a global transducer and let $w = w_1 \cdots w_n$ be the coding of the words $u_1, \ldots, u_n \in \Sigma^*$ such that $str(w_i) = u_i$. Assume that $\mathcal{A}$ has a successful run on w and let $\mathcal{A}(w) = w' = w'_1 \cdots w'_n \in (\Gamma \times \mathcal{D})^*$ where w'_i is a relabelling of w_i. To compute $str(w'_i)$ it suffices to know the word u_i and the pair (p, q) of states of the automaton $\mathcal{A}$ which are respectively the state of the automaton $\mathcal{A}$ before and after reading the word $\mathrm{mstr}(u_i)$ on the unique run on $\mathrm{mstr}(w)$. Among these, the pair (p, q) can be computed a finitary function that aggregates the set of all possible partial runs of $\mathcal{A}$ on each of the words $u_1, \ldots, u_n$ and hence an expression of fixed height can compute the word $str(w'_i)$.

▶ **Lemma 14.** *For each global transducer $\mathcal{A} : (\Sigma \times 2^Z)^* \to \Gamma^*$ and each $n \in \mathbb{N}$ there exist combinatorial expressions $e_1(\bar{X}), \ldots, e_n(\bar{X})$, where $\bar{X} = X_1 : \Sigma^*, \ldots, X_n : \Sigma^*$, of depth 5 such that for all n-tuple $\bar{u} = (u_1, \ldots, u_n)$ of words in Σ^* of identical even length $coding(e_1(\bar{u}), \ldots, e_n(\bar{u})) = \mathcal{A}(coding(\bar{u}))$.*

The above two lemmas can be generalised to a similar claim on cascades by induction (on the height of the cascade).

▶ **Lemma 15.** *For a cascade $\mathcal{C} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_k \rangle$ with input alphabet Σ, and each $n \in \mathbb{N}$ there exist combinatorial expressions $e_1(\bar{X}), \ldots, e_n(\bar{X})$, where $\bar{X} = X_1 : \Sigma^*, \ldots, X_n : \Sigma^*$, of depth atmost $7k$ such that for all n-tuple $\bar{u} = (u_1, \ldots, u_n)$ of words in Σ^* of identical even length $coding(e_1(\bar{u}), \ldots, e_n(\bar{u})) = \mathcal{C}(coding(\bar{u}))$.*

Next we define a data language B_n that corresponds to the property V_n.

For a word $w = a_1 a_2 \ldots a_l \in \{0, 1\}^*$ we let $pad(w) = 1a_1 1a_2 \cdots 1a_l$. We will also use *pad* as a binary gate. A *bridge* in a data word w is a sequence of positions along a path that consists of alternating class successor and global successor edges. Formally the sequence of positions $i_1, \ldots, i_n$ forms a bridge in w if there exists a sequence of successor and class successor edges $e_1, \ldots e_{n-1}$ in w such that for each $1 \le j < n$, $e_j = (i_j, i_{j+1})$ and for each $1 \le j < n - 1$, e_j is a successor edge iff e_{j+1} is a class successor edge. A bridge is a-labelled, for $a \in \Sigma$, if all the positions in the bridge are labelled by the letter a.

▶ **Definition 16.** Let $B_n \subseteq (\{0, 1\} \times \mathcal{D})^*$ be the set of all data words w such that w has a 1-labelled bridge $i_1, \ldots, i_{2n-1}$ (connected by a path of $2n-2$ edges), and

1. all positions to the left of i_1 are first positions of classes,
2. all positions to the right of i_{2n-1} are last positions of classes, and
3. the path corresponding to the bridge starts with a class successor edge.

Define the data language $B = \bigcup_{n=1}^{\infty} B_n$.

The language B_n is defined by the BMA formula (also in unary-DataLTL) of comp-height $2n + 1$,

$$\mathsf{F}^g\,(\mathsf{H}^g\mathsf{fst}^c \wedge (\mathtt{1X}^c\mathtt{1X}^g)^n\mathsf{G}^g\mathsf{lst}^c) \quad \text{where } \mathtt{1X}^g\varphi \text{ stands for the formula } (1 \wedge \mathtt{X}^g\varphi), \text{ and } \mathtt{1X}^c\varphi \text{ for } (1 \wedge \mathtt{X}^c\varphi). \tag{1}$$

Similarly the language B is defined by the BR formula

$$\mathsf{fst}^c\,\mathtt{U}^g\,(\mu x.(\mathtt{1X}^c\mathtt{1X}^g x \vee \mathtt{1X}^c\mathtt{1X}^g\mathsf{G}^g\mathsf{lst}^c)) \ . \tag{2}$$

▶ **Proposition 17.** *Let $(u_1, \ldots, u_n)$ be a tuple of words of identical length over the alphabet $\{0, 1\}$. Then the following are equivalent.*

1. $(u_1, \ldots, u_n) \in V_n$.

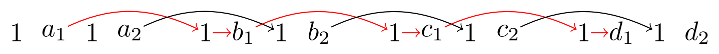
1 a_1 1 a_2 1 b_1 1 b_2 1 c_1 1 c_2 1 d_1 1 d_2

Figure 4 The data word w corresponding to the words $a_1a_2, b_1b_2, c_1c_2, d_1d_2$, and a bridge of length 7 in w.

2. *The data word $w = coding(pad(u_1), \ldots, pad(u_n))$ is in the language B_n.*

For a data language $L \subseteq (\Sigma \times \mathcal{D})^*$ we write $L^c = \{w \in (\Sigma \times \mathcal{D})^* \mid w \notin L\}$ for the complement of L. The data language $L \subseteq (\Sigma \times \mathcal{D})^*$ *separates* the data languages $L_1, L_2 \subseteq (\Sigma \times \mathcal{D})^*$ if $L_i \cap L = \emptyset$ and $L_{1-i} \subseteq L$ for some $i \in \{0, 1\}$. A cascade $\mathcal{C}$ (respectively a formula φ) separates the data languages L_1, L_2 if $L(\mathcal{C})$ (respectively $L(\varphi)$) separates L_1, L_2.

▶ **Lemma 18.** *If there is a cascade $\mathcal{C}$ of height k that separates the data languages $L_1 = B_n \cap n$-Codings, $L_2 = (B_n)^c \cap n$-Codings then there is a combinatorial expression of depth $7k + 4$ recognising the property V_n.*

Proof. Assume that $\mathcal{C}$ is a cascade of height k separating the languages L_1, L_2. Since cascades (of height k) are closed under complementation, without loss of generality assume that $L(\mathcal{C}) \supseteq L_1$ and $L(\mathcal{C}) \cap L_2 = \emptyset$. Therefore the cascade $\mathcal{C}$ produces an output on a data word n-Codings $\ni w \in (\{0, 1\} \times \mathcal{D})^*$ if and only if w is in the language B_n. Let $e_1(\bar{X}), \ldots, e_n(\bar{X})$, for $\bar{X} = X_1 : \{0, 1\}^*, \ldots, X_n : \{0, 1\}^*$, be the combinatorial expressions of depth at most $7k$, guaranteed by the Lemma 15 such that for all n-tuple $\bar{u} = (u_1, \ldots, u_n)$ of words in $\{0, 1\}^*$ of identical even length, $coding(e_1(\bar{u}), \ldots, e_n(\bar{u})) = \mathcal{C}(coding(\bar{u}))$.

Let $pad(\bar{X})$ stand for the vector of expressions $pad(X_1), \ldots, pad(X_n)$. We claim that *the expression*

$$e = \bigwedge (NE(e_1(pad(\bar{X})), \ldots, e_n(pad(\bar{X})), t(X_1, \ldots, X_n)) \ ,$$

where t is the expression from Example 10 for the alphabet $\{0, 1\}$ that checks if all the input words are of the same length, computes the property V_n. The expression e has depth at most $7k + 4$. To show the claim it is enough to verify that for a tuple $\bar{u} = (u_1, \ldots, u_n)$ of words from $\{0, 1\}^*$ of equal length, none of the words $v_1 = e_1(pad(\bar{u})), \ldots, v_n = e_n(pad(\bar{u}))$ is the empty word if and only if $\bar{u} \in V_n$. By Lemma 15, the words v_1 to v_n are nonempty iff $\mathcal{C}$ accepts the data word $w = coding(pad(\bar{u}))$. By assumption, the data word w is accepted by the cascade $\mathcal{C}$ iff $w \in B_n$. By Lemma 17, the data word w is in the language B_n iff $\bar{u}$ is in the property V_n. Hence the claim is proved. ◀

We are now ready for the main theorem;

▶ **Theorem 19** (Separation). *Let $N = 7k+4$.*

1. *The data languages $L_1 = B_{2^N+1} \cap (2^N+1)$-Codings and $L_2 = (B_{2^N+1})^c \cap (2^N+1)$-Codings are not separable by a formula in* BMA *of comp-height k.*
2. *The data language B_{2^N+1} is not definable by a formula in* BMA *of comp-height k.*
3. *Class of* BMA *definable languages form a hierarchy under composition height; more precisely for every k there exists a* BMA*-formula φ with comp-height k that has no equivalent formula of comp-height $k-1$.*
4. *The class of* BMA *definable languages is strictly subsumed by the class of* BR *definable languages.*

Proof. (**1.**) Proof by contradiction. Assume that the data languages L_1, L_2 are separable by a BMA formula φ of comp-height k. This implies that there is cascade of height k separating

L_1, L_2. By Lemma 18 there is an expression of depth N recognising the property V_{2^N+1}. This is in contradiction with Theorem 12.
(**2.**) Follows from (**1**).
(**3.**) From (**2.**) and the Equation (1), B_{2^N+1} is definable by a BMA formula of comp-height $2 \cdot (2^N+1)+1$ but not by any formula of comp-height k. Therefore (†) the set of languages defined by BMA^k is strictly contained in the set of languages defined by $\text{BMA}^{2\cdot(2^N+1)+1}$.

It only remains to separate the languages definable by BMA^k and the languages definable by BMA^{k+1}, for all k. We prove this claim by contradiction. Assume that ($\star$) there is some $m \in \mathbb{N}$ such that for every formula in BMA^{m+1} there is an equivalent formula in BMA^m. We claim that $f$or every formula in BMA^{m+2} there is an equivalent formula in BMA^m as well. To prove the claim, let $\chi = \psi(\varphi_1, \ldots, \varphi_n)$ be an arbitrary formula in BMA^{m+2} such that $\psi \in \text{BMA}^1$ and $\varphi_1, \ldots, \varphi_n \in \text{BMA}^{m+1}$. By assumption ($\star$) there exist formulas $\varphi'_1, \ldots, \varphi'_n \in \text{BMA}^m$ equivalent to the formulas $\varphi_1, \ldots, \varphi_n$ respectively. Therefore the formula $\chi' = \psi(\varphi'_1, \ldots, \varphi'_n)$ is equivalent to the formula χ and is in BMA^{m+1}. Appying the assumption ($\star$) again there is a formula $\chi'' \in \text{BMA}^m$ equivalent to χ' and hence also to χ, and hence the claim is proved. Extending this argument, by induction on k, it can be shown that for every formula in BMA^{m+k} there is an equivalent formula in BMA^m. This is in contradiction with the statement (†). Hence the statement is proved.
(**4.**) We claim that the data language B is not definable by any BMA formula. For the sake of contradiction, assume that there is a BMA formula φ of comp-height k recognising the language B and let $\mathcal{C}$ be the cascade of height k corresponding to φ. We claim that the cascade $\mathcal{C}$ separates the languages L_1 and L_2. Clearly by definition of the language B, $L_1 \subseteq B$. We need to show that $L_2 \cap B = \emptyset$ and it suffices to prove that for every $w \in (2^N+1)$-Codings if $w \in B$ then $w \notin (B_{2^N+1})^c$. Since any coding w in (2^N+1)-Codings either belongs to B_{2^N+1} or does not belong to B, it follows that if $w \in B$ then $w \notin (B_{2^N+1})^c$. Therefore the cascade $\mathcal{C}$ separates the languages L_1 and L_2 which contradicts (**1.**) and hence the claim follows. On the other hand, since B is definable by a formula in BR (Equation 2), the statement is proved.

◀

5 Conclusion

In this paper we studied the some fragments of μ-calculus over data words. We disclosed two fragments that are: the Bounded Reversal fragment (BR) and the Bounded Mode Alternation fragment (BMA) and proved they are separate. BR and BMA happen to form Boolean algebras making them very natural, and relatively expressive logics over data words. We also establish the relationship with earlier logics like FO^2 or DataLTL.

References

1 Henrik Björklund and Thomas Schwentick. On notions of regularity for data languages. *Theor. Comput. Sci.*, 411(4-5):702–715, 2010.

2 M. Bojańczyk and S. Lasota. An extension of data automata that captures xpath. In *Logic in Computer Science (LICS), 2010*, pages 243–252, July 2010.

3 Mikołaj Bojańczyk. Data monoids. In *STACS*, pages 105–116, 2011.

4 Mikołaj Bojańczyk, Claire David, Anca Muscholl, Thomas Schwentick, and Luc Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27, 2011.

5 Thomas Colcombet, Clemens Ley, and Gabriele Puppis. On the use of guards for logics with data. In *MFCS*, volume 6907 of *LNCS*, pages 243–255. Springer, 2011.

6 Thomas Colcombet and Amaldev Manuel. Generalized data automata and fixpoint logic. In *FSTTCS 2014*, volume 29 of *LIPIcs*, pages 267–278, 2014.

7 Thomas Colcombet and Amaldev Manuel. Combinatorial expressions and lower bounds. In *STACS*, 2015 (To appear).

8 S. Demri, D. Figueira, and M. Praveen. Reasoning about data repetitions with counter systems. In *Logic in Computer Science (LICS), 2013*, pages 33–42, June 2013.

9 Stéphane Demri and Ranko Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), April 2009.

10 Kousha Etessami, Moshe Y. Vardi, and Thomas Wilke. First-order logic with two variables and unary temporal logic. *Inf. Comput.*, 179(2):279–295, 2002.

11 D. Figueira. Alternating register automata on finite data words and trees. *Logical Methods in Computer Science*, 8(1), 2012.

12 D. Figueira. Decidability of downward XPath. *ACM Transactions on Computational Logic*, 13(4), 2012.

13 O. Grumberg, O. Kupferman, and S. Sheinvald. Variable automata over infinite alphabets. In *Language and Automata Theory and Applications*, pages 561–572. Springer, 2010.

14 M. Jurdziński and R. Lazic. Alternating automata on data trees and xpath satisfiability. *ACM Trans. Comput. Log.*, 12(3):19, 2011.

15 Michael Kaminski and Nissim Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

16 Ahmet Kara, Thomas Schwentick, and Thomas Zeume. Temporal logics on words with multiple data values. In *FSTTCS*, volume 8 of *LIPIcs*, pages 481–492, 2010.

17 L. Libkin and D. Vrgoc. Regular expressions for data words. In *LPAR*, volume 7180 of *Lecture Notes in Computer Science*, pages 274–288. SPRINGER, 2012.

18 A. Manuel, A. Muscholl, and G. Puppis. Walking on data words. In *Computer Science Theory and Applications*, volume 7913 of *LNCS*, pages 64–75. Springer, 2013.

19 F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004.

20 Igor Walukiewicz. Completeness of kozen's axiomatisation of the propositional μ-calculus. *Information and Computation*, 157(1–2):142 – 182, 2000.

A Proof of Theorem 5

In this section we prove the following inclusions,

$$\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c 1) = \mathrm{uDataLTL} \subsetneq \mathrm{DataLTL} \subsetneq \mathrm{BMA} \subseteq \mathrm{BR} \subsetneq \nu \subseteq \text{Data Automata}.$$

The inclusion of unary-DataLTL in DataLTL is straight-forward since the modalities $\mathtt{F}^c, \mathtt{F}^g, \mathtt{P}^g, \mathtt{P}^c$ are expressible in terms of $\mathtt{U}^g, \mathtt{S}^g, \mathtt{U}^c, \mathtt{S}^c$ in the standard way; for instance $\mathtt{F}^c\varphi \equiv \top\, \mathtt{U}^c\, \varphi$.

The inclusion of DataLTL in BMA follows from the Example 1 since every modality of DataLTL is expressible in BMA and BMA is closed under composition.

The strictness of the two inclusions mentioned above, can be inferred from the strictness of inclusion of the corresponding logics on words over a finite alphabet in the following way. Words over the alphabet Σ can be seen as data words over the alphabet $\Sigma \times \mathcal{D}$ such that all data values appearing in the data word are the same, i.e. that satisfy the formula $\chi = \mathtt{G}^g(\mathcal{S} \vee \mathsf{lst}^g)$. Consider the set of all data words of even length that satisfy the formula χ. We claim that this set (call it L) is not definable in DataLTL. Clearly L is definable in BMA by the formula $\mu x.(\mathtt{X}^g\mathsf{lst}^g \vee \mathtt{X}^g\mathtt{X}^g x) \wedge \chi$. We claim that L is not definable in DataLTL. For the sake of contradiction, assume that there is a formula φ defining L, then the LTL formula φ' obtained by replacing all class modalities by the corresponding global ones, the modality $\mathcal{S}$ by $\neg\mathsf{lst}^g$, and the modality $\mathcal{P}$ by $\neg\mathsf{fst}^g$ defines the set of all words over Σ of even length. But this is a contradiction since words of even length are not LTL definable. Thus we conclude that L is definable in BMA but not in DataLTL. The proof of separation of DataLTL and uDataLTL is similar; one takes a language that separates LTL and unary LTL, for instance the language $(c^*ac^*b)^*$, and repeats the same argument.

Finally, the inclusion $\nu \subseteq$ Data Automata is shown in [6]. Next we prove the remaining inclusions.

A.1 $\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c 1) = \mathrm{uDataLTL}$

We aim at proving the equality $\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c 1) = \mathrm{uDataLTL}$. The nontrivial direction is the one from FO^2 formulas to modal formulas. Here the only difficulty is that FO^2 formulas can quantify over 'positions not in class' (for instance $\exists y(x < y \wedge x \not\sim y \wedge b(y))$) whereas in the temporal logic there is no such mechanism available. Our first lemma deals with this situation and shows that such cases can be handled using our modalities.

First we define the modalities $\mathtt{fF}^{\not\sim}$ (*far-future not in class*) and $\mathtt{dP}^{\not\sim}$ (*deep-past not in class*) as,

$$\begin{aligned} w, i \models \mathtt{fF}^{\not\sim}\varphi \quad &\Leftrightarrow \quad \exists j > i+1 \text{ such that } i \not\sim j \text{ and } w, j \models \varphi \\ w, i \models \mathtt{dP}^{\not\sim}\varphi \quad &\Leftrightarrow \quad \exists j < i-1 \text{ such that } i \not\sim j \text{ and } w, j \models \varphi \end{aligned}$$

▶ **Lemma 20.** *The modalities $\mathtt{fF}^{\not\sim}$ and $\mathtt{dP}^{\not\sim}$ are expressible using the modalities $\{\mathtt{X}^g, \mathtt{X}^c, \mathtt{Y}^g, \mathtt{Y}^c, \mathtt{F}^c, \mathtt{F}^g, \mathtt{P}^g, \mathtt{P}^c\}$ over data words.*

Proof. We only do the case of $\mathtt{fF}^{\not\sim}$. The case of $\mathtt{dP}^{\not\sim}$ is symmetric. Assume we are given a formula $\mathtt{fF}^{\not\sim}\varphi$. Let k be the last position where φ is true. Obviously it is the unique position where $\varphi_{last} = \varphi \wedge \neg\mathtt{F}^g\varphi$ is true. A position i satisfies $\mathtt{fF}^{\not\sim}\varphi$ if and only if one of the following scenarios hold;

1. $k > i+1$ and $k \not\sim i$,
2. $k \sim i$ and there is a $j > i+1$ such that j satisfies φ and $j \not\sim k$.

The first scenario holds if the formula $\mathtt{X}^g\mathtt{X}^g\mathtt{F}^g\varphi_{last} \wedge \neg\mathtt{F}^c\varphi_{last}$ is true at position i. (Note that $\mathtt{F}^g$ evaluates a formula on all positions in the future including the current position, hence $\mathtt{X}^g\mathtt{X}^g\mathtt{F}^g\varphi_{last}$). The second scenario holds if the formula $\mathtt{F}^c\varphi_{last} \wedge \mathtt{X}^g\mathtt{X}^g\mathtt{F}^g(\varphi \wedge \neg\mathtt{F}^c\varphi_{last})$ holds at position i. Hence $\mathtt{fF}^{\not\sim}\varphi$ is equivalent to the formula

$$\Psi \equiv (\mathtt{X}^g\mathtt{X}^g\mathtt{F}^g\varphi_{last} \wedge \neg\mathtt{F}^c\varphi_{last}) \vee (\mathtt{F}^c\varphi_{last} \wedge \mathtt{X}^g\mathtt{X}^g\mathtt{F}^g(\varphi \wedge \neg\mathtt{F}^c\varphi_{last})).$$

◀

▶ **Corollary 21.** *The modalities $\mathtt{F}^{\not\sim}$ (future not in class) and $\mathtt{P}^{\not\sim}$ (past not in class) defined as*

$$\begin{aligned} w, i \models \mathtt{F}^{\not\sim}\varphi &\Leftrightarrow \exists j > i \text{ such that } i \not\sim j \text{ and } w, j \models \varphi \\ w, i \models \mathtt{P}^{\not\sim}\varphi &\Leftrightarrow \exists j < i \text{ such that } i \not\sim j \text{ and } w, j \models \varphi \\ \mathtt{G}^{\not\sim}\varphi &\Leftrightarrow \neg\mathtt{F}^{\not\sim}\neg\varphi \\ \mathtt{H}^{\not\sim}\varphi &\Leftrightarrow \neg\mathtt{P}^{\not\sim}\neg\varphi \end{aligned}$$

is definable in DLTL over data words.

Proof. Define $\mathtt{F}^{\not\sim}\varphi \equiv (\neg\mathcal{S} \wedge \mathtt{X}^g\varphi) \vee \mathtt{fF}^{\not\sim}\varphi$ and $\mathtt{P}^{\not\sim}\varphi \equiv (\neg\mathcal{P} \wedge \mathtt{Y}^g\varphi) \vee \mathtt{dP}^{\not\sim}\varphi$. ◀

Next we show that for every FO^2 formula with one free variable there is a corresponding uDataLTL formula and vice versa, thus proving the equality $\mathrm{FO}^2(\Sigma, <, +1, \sim, +^c1) =$ uDataLTL.

▶ **Lemma 22.** *1. For every uDataLTL formula φ there is a $\mathrm{FO}^2\,(\Sigma, <, +1, \sim, +^c1)$ formula $\varphi'(x)$ such that $w, i \models \varphi$ if and only if $w, i \models \varphi'(x)$.*
2. Similarly, for every $\mathrm{FO}^2\,(\Sigma, <, +1, \sim, +^c1)$ formula $\varphi(x)$ there is a uDataLTL formula φ' such that $w, i \models \varphi'$ if and only if $w, i \models \varphi(x)$.

Proof. (**1.**) Follows simply from the fact that the modalities used in uDataLTL are expressible in $\mathrm{FO}^2\,(\Sigma, <, +1, \sim, +^c1)$ and we use the obvious analogue of the standard translation from modal logic to two-variable first order logic. For variables $\{x_0, x_1\}$ define the translation operators $\langle\ \rangle^{x_i}$, $i \in \{0, 1\}$ from uDataLTL formulas to FO^2 formulas with a free variable x_i as follows: (we omit the past modalities whose translations are symmetric)

$$\begin{aligned} \langle a\rangle^{x_i} &= a(x_i) \\ \langle\mathcal{S}\rangle^{x_i} &= \exists x_{1-i}\,(x_{1-i} = x_i + 1 \wedge x_{1-i} = x_i +^c 1) \\ \langle\mathcal{P}\rangle^{x_i} &= \exists x_{1-i}\,(x_i = x_{1-i} + 1 \wedge x_i = x_{1-i} +^c 1) \\ \langle\neg\varphi\rangle^{x_i} &= \neg\langle\varphi\rangle^{x_i} \\ \langle\varphi_1 \vee \varphi_2\rangle^{x_i} &= \langle\varphi_1\rangle^{x_i} \vee \langle\varphi_2\rangle^{x_i} \\ \langle\mathtt{X}^g\varphi\rangle^{x_i} &= \exists x_{1-i}(x_{1-i} = x_i + 1 \wedge \langle\varphi\rangle^{x_{1-i}}) \\ \langle\mathtt{X}^c\varphi\rangle^{x_i} &= \exists x_{1-i}(x_{1-i} = x_i +^c 1 \wedge \langle\varphi\rangle^{x_{1-i}}) \\ \langle\mathtt{F}^g\varphi\rangle^{x_i} &= \exists x_{1-i}(x_{1-i} \geq x_i \wedge \langle\varphi\rangle^{x_{1-i}}) \\ \langle\mathtt{F}^c\varphi\rangle^{x_i} &= \exists x_{1-i}(x_{1-i} \geq x_i +^c 1 \wedge \langle\varphi\rangle^{x_{1-i}}) \end{aligned}$$

(**2.**) For convenience, we define the abbreviations $x \ll y$ and $x \ll^c y$ for $x < y \wedge x{+}1 \neq y$ and $x \sim y \wedge x < y \wedge x{+}^c1 \neq y$ respectively.

We intend to prove that for every $\mathrm{FO}^2\,(\Sigma, <, +1, \sim, +^c1)$ formula $\varphi(x)$ there is a uDataLTL formula φ' such that $w, i \models \varphi'$ if and only if $w, i \models \varphi(x)$. The proof idea is standard (see

[10]). The proof is by simultaneous induction on the structure of the formulas as well as on their quantifier depth. When $\varphi(x)$ is $a(x)$ then φ' is simply a. When $\varphi(x)$ is of the form $\varphi_1(x) \vee \varphi_2(x)$ (or $\neg\varphi_1(x)$), using induction hypothesis, we define φ' as $\varphi'_1 \vee \varphi'_2$ (or $\neg\varphi'_1$). The remaining cases are when $\varphi(x)$ is of the form $\exists x.\varphi_1(x)$ or $\exists y.\varphi_1(x,y)$. Both cases are identical upto a renaming of variables. So it is enough to consider only $\exists y.\varphi(x,y)$. We write $\varphi(x,y)$ in disjunctive normal form and distribute the existential quantifier over the disjunctions to obtain a formula of the form $\bigvee_i \exists y.\varphi_i(x,y)$ where each $\varphi_i(x,y)$ is of the form $\alpha_i(x) \wedge \beta_i(y) \wedge \delta_i(x,y) \wedge \gamma_i(x,y)$ in which $\alpha_i(x), \beta_i(y)$ are formulas with only one free variable, $\delta_i(x,y) \in \Delta(x,y)$, and $\gamma_i(x,y) \in \Gamma(x,y)$, where the sets $\Delta(x,y)$ and $\Gamma(x,y)$ are,

$$\Delta(x,y) = \{y \ll x,\ y+1 = x,\ x = y,\ x+1 = y,\ x \ll y\},$$

$$\Gamma(x,y) = \{y \ll^c x,\ y{+}^c 1 = x,\ x \not\sim y,\ x{+}^c 1 = y,\ x \ll^c y\}.$$

Note that writing each conjuct φ_i in this form might require replacing subformulas in φ_i which are negations of formulas in $\Delta(x,y)$ by an equivalent formula consisting of disjunctions of formulas from $\Delta(x,y)$ (and further distributing these disjunctions in the conjunct). Let us observe that it is enough to define a translation for each of the disjunct of the form $\varphi(x) \equiv \exists y.\, \alpha(x) \wedge \beta(y) \wedge \delta(x,y) \wedge \gamma(x,y)$. Since the quantifier depth of $\alpha(x)$ and $\beta(y)$ are strictly less than the quantifier depth of the formula $\varphi(x)$, by induction hypothesis we have the DataLTL formulas α' and β' that are equivalent to $\alpha(x)$ and $\beta(y)$. We define the translation below.

Consider the case when $\gamma(x,y)$ is $x \not\sim y$. Then the translations are listed below.

$\delta(x,y)$	φ'
$x = y$	false
$x \ll y$	$\alpha' \wedge \mathtt{fF}^{\not\sim}\beta'$
$x+1 = y$	$\alpha' \wedge \neg\mathcal{S} \wedge \mathtt{X}^g\beta'$
$y+1 = x$	$\alpha' \wedge \neg\mathcal{P} \wedge \mathtt{Y}^g\beta'$
$y \ll x$	$\alpha' \wedge \mathtt{dP}^{\not\sim}\beta'$

The rest of the cases are symmetric and hence we treat only the cases when $x \leq y$.

Assume $\delta(x,y) = x \ll y$. Then $\varphi(x,y)$ is satisfiable only when $\gamma(x,y)$ is $x{+}^c1 = y$ or $x \ll^c y$ and we define the respective translations as $\alpha \wedge \mathtt{X}^c\beta' \wedge \neg\mathcal{S}$ and $\alpha \wedge \mathtt{X}^c\mathtt{X}^c\mathtt{F}^c\beta'$.

When $\delta(x,y)$ is $x+1 = y$, $\varphi(x,y)$ is satisfiable only when $\gamma(x,y)$ is $x{+}^c1 = y$ and we define the translation as $\alpha \wedge \mathtt{X}^c\beta' \wedge \mathcal{S}$. This completes the construction. ◀

A.2 BMA $\subseteq$ BR

In this subsection we show that,

▶ **Lemma 23.** *For every formula φ in* BMA *of comp-height k there is an equivalent formula φ' in* BR *of comp-height $k+1$.*

In the proof we will use the separation property of μ-calculus on words. By μ-calculus on finite words, we mean the set of formulas φ described the following syntax:

$$\varphi := x \mid A \mid \neg A \mid \mathtt{X}\,\varphi \mid \mathtt{Y}\,\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mu x.\varphi \mid \nu x.\varphi$$
$$\text{where} \quad A := p \in \mathit{Prop} \mid \mathsf{first} \mid \mathsf{last}$$

The modalities X and Y evaluates the argument formula on the next and previous positions respectively, while the zeroary modalities first and last hold at the first position and last position respectively. Rest of the constructs have the intended semantics. A formula φ is *right* if it does not use the modality $\mathtt{Y}$. Similarly a formula is *left* if it does not use the modality $\mathtt{X}$. The below fact follows from the correspondence between μ-calculus formulas on words and finite state automata.

▶ **Fact 24.** *Any μ-calculus formula over words is equivalent to a Boolean combination of left and right formulas.*

Now we continue the proof of the Lemma 23.

Proof. We prove the following claim by induction, *for every formula of φ in* BMA *of comp-height k there is an equivalent formula φ' which is a Boolean combination of formulas in* BR *of comp-height k.* Since a Boolean combination of BR formulas of comp-height k has comp-height $k+1$ the lemma follows.

For the base case let φ be in $\mathit{formulas}\,(\{\mathtt{X}^g, \mathtt{Y}^g\}) \cup \mathit{formulas}\,(\{\mathtt{X}^c, \mathtt{Y}^c\})$ (of comp-height 1). Consider the case when φ is in $\mathit{formulas}\,(\{\mathtt{X}^g, \mathtt{Y}^g\})$. Let w be a data word. One can think of φ as a formula evaluated over a word $\mathrm{mstr}(w)$ over the alphabet $P = 2^{\mathit{Prop}(\varphi)} \times 2^Z$ where $\mathit{Prop}(\varphi)$ denotes the propositional variables used in φ. Seen this way the modalities $\mathtt{X}^g$ and $\mathtt{Y}^g$ is synonymous with $\mathtt{X}$ and $\mathtt{Y}$ over the word $\mathrm{mstr}(w)$ (and fst^g and lst^g are equivalent to first and last). Therefore we can apply Fact 24 to obtain a formula φ' that is a Boolean combination of formulas using only $\mathtt{X}^g$ and formulas using only $\mathtt{Y}^g$. Next consider the case when φ is a formula in $\mathit{formulas}\,(\{\mathtt{X}^c, \mathtt{Y}^c\})$. Let φ' be the formula obtained from φ by replacing $\mathtt{X}^c$ by $\mathtt{X}$, $\mathtt{Y}^c$ by $\mathtt{Y}$, fst^c by first, and lst^c by last. It is straightforward to see that $(\star)$ for a data word w and a position i in w, it is the case that $w, i \models \varphi$ if and only if $\mathrm{mstr}(w|_S), i' \models \varphi'$ where S is the class of the position i, and i' is the position in $\mathrm{mstr}(w|_S)$ corresponding to the position i in w. Let ψ' be a Boolean combination of left and right formulas given by Fact 24, that is equivalent to the formula φ' and let ψ be the formula obtained from ψ' by replacing $\mathtt{X}$ by $\mathtt{X}^c$, and $\mathtt{Y}$ by $\mathtt{Y}^c$, first by fst^c, and last by lst^c. By claim $(\star)$ ψ is equivalent to φ and is in the desired form. This completes the base case.

For the inductive step, let $\varphi = \psi(\varphi_1, \ldots, \varphi_k)$ be a BMA formula of comp-height $k+1$ where $\psi(x_1, \ldots, x_k) \in \mathit{formulas}\,(\{\mathtt{X}^g, \mathtt{Y}^g\}) \cup \mathit{formulas}\,(\{\mathtt{X}^c, \mathtt{Y}^c\})$ and $\varphi_1, \ldots, \varphi_k$ are BMA formulas of comp-height k. Using induction hypothesis we obtain $\varphi_1', \ldots, \varphi_k'$ which are Boolean combinations of BR formulas of comp-height k and are equivalent to $\varphi_1, \ldots, \varphi_k$ respectively. Repeating the previous argument we also obtain $\psi'(x_1, \ldots, x_k) \in \mathsf{Bool}(\mathit{formulas}\,(\{\mathtt{X}^c, \mathtt{X}^g\}) \cup \mathit{formulas}\,(\{\mathtt{Y}^c, \mathtt{Y}^g\}))$ equivalent to $\psi(x_1, \ldots, x_k)$. To conclude observe that $\psi'(\varphi_1', \ldots, \varphi_k')$ is a Boolean combination of BR formulas of comp-height at most $k+2$. ◀

A.3 BR $\subseteq$ ν-Fragment

Next we show that every 'guarded' formula in BR has a unique fixpoint and hence such formulas can be written only using greatest fixpoints. This, in conjunction with that fact that any formula of the μ-calculus can be written in guarded form, implies that BR is subsumed by the ν-fragment. Intuitively, guarded μ-calculus formulas represent (alternating parity) automata whose transition strictly moves left or right on each transition. Since data words are finite and BR formulas make only a bounded number of reversals, all runs of the corresponding automata have to terminate, this implies that the parities (which in turn determines the kind of fixpoint) of the states are irrelevant.

First we recall the definition of guarded formulas. We say a variable x in $\lambda x.\varphi(x)$ is *guarded* if each occurrence of x in $\varphi(x)$ is in the scope of some modality. We say a formula φ is *guarded* if each bound variable in φ is guarded. Next we show that every formula can be written in guarded form. The proof below is a repetition from [20].

▶ **Lemma 25.** *Every formula is equivalent to a formula which is furthermore guarded.*

Proof. Proof is by induction on the structure of the formula. The atomic, Boolean and modal cases are straightforward. The non-trivial case is when the formula is of the form $\lambda x.\varphi(x)$. Assume $\lambda x.\varphi(x)$ is unguarded and $\varphi(x)$ is guarded. We can furthermore assume that all unguarded occurrences of x are outside of any subformula $\theta y.\psi(x,y)$ of $\varphi(x)$, otherwise in $\varphi(x)$ we substitute for $\theta y.\psi(x,y)$ the equivalent formula $\psi(x, \theta y.\psi(x,y))$ which yields the desired form. Next we write $\varphi(x)$ is conjunctive normal form to obtain a formula of the form

$$\lambda x.(x \vee \alpha(x)) \wedge \beta(x),$$

where $\alpha(x)$ and $\beta(x)$ are guarded. The claim follows from the observations below,

$$\mu x.(x \vee \alpha(x)) \wedge \beta(x) \equiv \mu x.\alpha(x) \wedge \beta(x)\,,$$

and

$$\nu x.(x \vee \alpha(x)) \wedge \beta(x) \equiv \nu x.\beta(x)\,.$$

◀

▶ **Lemma 26.** *Let $\varphi(x,\bar{y})$ be a formula such that the only unary modalities it uses are $\mathtt{Y}^g, \mathtt{Y}^c$ (respectively $\mathtt{X}^g, \mathtt{X}^c$) and furthermore any free occurrence of x appears in the scope of at least k nested modalities. Then for any data word w and valuation $S_1, \ldots, S_l$ of $\bar{y} = y_1, \ldots, y_l$, and S of x, and for all $i < k$ (respectively $i > n-k$)*

$$w[\ell(\bar{y}) := \bar{S}, \ell(x) = S], i \models \varphi \quad \Leftrightarrow \quad w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset], i \models \varphi\ .$$

Proof. Since both claims are symmetric it is enough to prove one of them. We treat the case when the only unary modalities $\varphi(x,\bar{y})$ uses are $\mathtt{Y}^c$ and $\mathtt{Y}^g$. Without loss of generality assume that x is not a bound variable in $\varphi(x,\bar{y})$ (otherwise rename the bound occurrences of x). We proceed by an induction on the pair (k,i) ordered lexicographically (for all $i \geq k$ the claim holds trivially); For the base case when $k = 1$, the claim is vacuously true. For the inductive step assume the claim is true for pairs (k', i') where $k' < k$ or, $k' = k$ and $i' < i$. Let $\varphi(x,\bar{y})$ be a formula in which x appears with in the scope of $k+1$ nested modalities. We do an induction on the structure of the formula. Let $\varphi(x,\bar{y})$ is of the form $\mathtt{M}\psi(x,\bar{y})$ where $\mathtt{M} \in \{\mathtt{Y}^g, \mathtt{Y}^c\}$. We do a case analysis on $\mathtt{M}$. Assume $\mathtt{M}$ is $\mathtt{Y}^g$ (the case when $\mathtt{M}$ is $\mathtt{Y}^c$ being analogous) then

$$\begin{aligned}
&w[\ell(\bar{y}) := \bar{S}, \ell(x) = S], i \models \mathtt{M}\psi(x,\bar{y}) \\
&\qquad \Leftrightarrow w[\ell(\bar{y}) := \bar{S}, \ell(x) = S], i-1 \models \psi(x,\bar{y}) && \text{(By defn. of } \mathtt{Y}^g) \\
&\qquad \Leftrightarrow w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset], i-1 \models \psi(x,\bar{y}) && (i < k \Rightarrow i-1 < k-1, \text{ hence by IH}) \\
&\qquad \Leftrightarrow w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset], i \models \mathtt{M}\psi(x,\bar{y})
\end{aligned}$$

The Boolean cases are straightforward. Next assume $\varphi(x,\bar{y})$ is of the form $\theta y_j.\psi(x,\bar{y})$ ($\theta \in \{\mu, \nu\}$). We have to show that

$$w[\ell(\bar{y}) := \bar{S}, \ell(x) = S], i \models \theta y_j.\psi(x,\bar{y}) \quad \Leftrightarrow \quad w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset], i \models \theta y_j.\psi(x,\bar{y})\ .$$

By induction hypothesis (on the structure of the formula)

$$w[\ell(\bar{y}) := \bar{S}, \ell(x) = S], i \models \psi(x, \bar{y}) \quad \Leftrightarrow \quad w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset], i \models \psi(x, \bar{y}) \ .$$

Hence S_j is a pre-fixpoint (*resp.* post-fixpoint) of $\psi(x, \bar{y})$ on $w[\ell(\bar{y}) := \bar{S}, \ell(x) = S]$ if and only if it is a pre-fixpoint (*resp.* post-fixpoint) of $\psi(x, \bar{y})$ on $w[\ell(\bar{y}) := \bar{S}, \ell(x) = \emptyset]$. Hence the claim is proved by Knaster-Tarski theorem. This concludes the induction. ◀

Next we state our final lemma,

▶ **Lemma 27.** *Every* BR*-formula is equivalent to a formula of the ν-fragment over data words.*

Proof. This is done in two steps. The first step is to transform the formula in BR to an equivalent one that is furthermore guarded. This is achieved by Lemma 25. In the second step we turn every subformula of the form $\mu x.\varphi(x, \bar{y})$ into $\nu x.\varphi(x, \bar{y})$. We claim that the resulting formula is equivalent to the original one. Thanks to Lemma 25, we only have to prove the correctness of the second step, which amounts to proving that *(Claim $\star$) given a guarded* BR*-formula, it is equivalent over all data words to the formula in which each μ-fixpoint is turned into a ν-fixpoint.*

Observe first that it is sufficient to prove ($\star$) for formulas in *formulas* $(\{\mathtt{X}^c, \mathtt{X}^g\})$. Indeed, from this result, by symmetry, it also holds for formulas in *formulas* $(\{\mathtt{Y}^c, \mathtt{Y}^g\})$. Note that, given formulas $\phi(x), \phi'(x), \psi$ such that $\phi(x)$ and $\phi'(x)$ are equivalent over all data words, then the same holds for the substitutions $\phi(\psi)$ and $\phi'(\psi)$. Since formulas in BR are obtained from formulas in *formulas* $(\{\mathtt{X}^c, \mathtt{X}^g\})$ and *formulas* $(\{\mathtt{Y}^c, \mathtt{Y}^g\})$ via inductive substitution, this implies ($\star$) for all formulas in BR.

Hence, what remains to be shown is that ($\star$) holds for a formula in $\psi \in$ *formulas* $(\{\mathtt{X}^c, \mathtt{X}^g\})$. Observe that by induction on the structure of the formula it is enough to verify that for each guarded formula $\psi = \mu x.\varphi(x, \bar{y}) \in$ *formulas* $(\{\mathtt{X}^c, \mathtt{X}^g\})$ and for every data word w (of length n) and valuation $S_1, \ldots, S_k$ (all of them subsets of $[n]$) of $\bar{y} = y_1, \ldots, y_k$,

$$[\![\nu x.\varphi(x, \bar{y})]\!]_{w'} \subseteq [\![\mu x.\varphi(x, \bar{y})]\!]_{w'}$$

where $w' = w[\ell(y_1) := S_1, \ldots, \ell(y_k) := S_k]$, since the other inclusion follows from the fact that the least fixpoint is always included in the greatest fixpoint. This reduces to showing that

$$w', i \models \nu x.\varphi(x, \bar{y}) \Rightarrow w', i \models \mu x.\varphi(x, \bar{y})$$

This is exhibited by the following calculation,

$$\begin{aligned} w', i \models \nu x.\varphi(x, \bar{y}) &\Leftrightarrow w', i \models \varphi(\nu x.\varphi(x, \bar{y}), \bar{y}) && \text{(By fixpoint iteration)} \\ &\Leftrightarrow w', i \models \varphi^{n+1}(\nu x.\varphi(x, \bar{y}), \bar{y}) \\ &\Rightarrow w', i \models \varphi^{n+1}(\bot, \bar{y}) && \text{(By Lemma 26)} \\ &\Rightarrow w', i \models \mu x.\varphi(x, \bar{y}) && \text{(By Knaster-Tarski theorem)} \end{aligned}$$

◀

Let us note that since the class of languages definable by the ν-fragment is not closed under complement while the class of languages definable by BR is closed under complement, it follows that BR is strictly less expressive than the ν-fragment over data words. This concludes the proof the Theorem 5.

B Proof of Proposition 7

In this section we prove the equivalence between BMA and the cascades of class and global transducers. The idea is to lift the correspondence between μ-calculus on words and finite state automata to the case of BMA and cascades (See Subsection A.2) for the definition of μ-calculus on words that we use here).

▶ **Lemma 28.** *Let Γ_1 and Γ_2 be two disjoint alphabets and let $\mathcal{A}_1 : \left(\Sigma \times 2^Z\right)^* \to \Gamma_1^*$ and $\mathcal{A}_2 : \left(\Sigma \times 2^Z\right)^* \to \Gamma_2^*$ be two global transducers (resp. class transducers). Then there exists a global transducer (resp. class transducer) $\mathcal{A} : \left(\Sigma \times 2^Z\right)^* \to (2^{\Gamma_1 \cup \Gamma_2})^*$ such that for every data word $w \in (\Sigma \times \mathcal{D})^*$, if $\mathcal{A}_1(w) = (a_1, d_1) \cdots (a_n, d_n)$ and $\mathcal{A}_2(w) = (a'_1, d_1) \cdots (a'_n, d_n)$ then $\mathcal{A}(w) = (\{a_1, a'_1\}, d_1) \cdots (\{a_n, a'_n\}, d_n)$.*

Proof. Consider $\mathcal{A}_1$ and $\mathcal{A}_2$ as word transducers and by product construction obtain the word transducer $\mathcal{A} : \left(\Sigma \times 2^Z\right)^* \to (2^{\Gamma_1 \cup \Gamma_2})^*$ such that for all words $w \in \left(\Sigma \times 2^Z\right)^*$ if $\mathcal{A}_1(w) = a_1 \cdots a_n$ and $\mathcal{A}_2(w) = a'_1 \cdots a'_n$ then $\mathcal{A}(w) = \{a_1, a'_1\} \cdots \{a_n, a'_n\}$. The transducer $\mathcal{A}$ satisfy the claim. ◀

▶ **Fact 29.** *Given a μ-calculus formula φ over words there is a non-deterministic finite state functional transducer $\mathcal{A}_\varphi$ such that given any word w the automaton $\mathcal{A}_\varphi$ outputs $\{\varphi\}$ (resp. $\emptyset$) exactly on those positions where φ is true (resp. false).*

▶ **Fact 30.** *Let L be a regular language over the alphabet Σ. Then there exist μ-calculus formulas over words $\varphi_{\mathtt{X}}$ and $\varphi_{\mathtt{Y}}$ such that*

1. *$\varphi_{\mathtt{X}}$ does not use the modality $\mathtt{Y}$ and $\varphi_{\mathtt{Y}}$ does not use the modality $\mathtt{X}$, and*
2. *for all words $w \in \Sigma^*$, $w, 1 \models \varphi_{\mathtt{X}}$ iff $w \in L$, and $w, |w| \models \varphi_{\mathtt{Y}}$ iff $w \in L$.*

▶ **Lemma 31.** *Given a letter-to-letter transducer $\mathcal{A} : \Sigma^* \to \Sigma'^*$ and a letter $a' \in \Sigma'$ there is a formula $\varphi_{a'}$ such that for any word $w \in \Sigma^*$ and a position i in w, it is the case that $w, i \models \varphi_{a'}$ if and only if there is an output word (unique in the case of functional transducers) $a'_1 a'_2 \ldots a'_n$ of the transducer $\mathcal{A}$ on w such that $a'_i = a'$.*

Proof. Assume the automaton $\mathcal{A}$ has states Q, initial state q_0 and set of final states $F \subseteq Q$ and transition relation $\Delta \subseteq Q \times \Sigma \times Q \times \Sigma'$. On a position i of an input word $w = a_1 \ldots a_n$, the automaton outputs the symbol a' if there is a transition $(p, a_i, q, a') \in \Delta$ such that

1. there is a run of the automaton $\mathcal{A}$ on the word $a_1 \ldots a_{i-1}$ starting from initial state q_0 and ending in state p, and
2. there is a run of the automaton $\mathcal{A}$ on the word $a_{i+1} \ldots a_n$ starting from the state q and ending in a final state.

For a state $p \in Q$, let ψ_p be a μ-calculus formula on words using only the modality $\mathtt{Y}$ expressing the regular property that the automaton $\mathcal{A}$ has a run starting from the initial state and ending in the state p on the prefix defined by the current position (excluding the current position) as guaranteed by the Fact 30. Similarly, for $p \in Q$, let χ_p a formula on words using only the modality $\mathtt{X}$ expressing the regular property that the automaton $\mathcal{A}$ has a run starting from the state p and ending in a final state on the suffix defined by the current position (excluding the current position) (again, ensured by the Fact 30). Then the desired formula $\varphi_{a'}$ is

$$\bigvee_{(p,a,q,a') \in \Delta} (\psi_p \wedge a \wedge \chi_q).$$

◀

Next we prove the Proposition.

Let L be a data set of data words. Then the following statements are equivalent.

1. L is definable by a formula in BMA of comp-height k.
2. L is recognisable by a cascade of height k.

Proof. ($1 \Rightarrow 2$) Observe that it is sufficient to prove that ($\star$) *for every formula φ in formulas* $(\{\mathtt{X}^g, \mathtt{Y}^g\})$ *on data words there is a global transducer $\mathcal{A}$ outputting $\{\varphi\}$ (resp. $\emptyset$) exactly at those positions where φ does (resp. not) hold.* By Lemma 28 the claim ($\star$) holds for a finite set of formulas. By symmetry a similar claim holds for φ in *formulas* $(\{\mathtt{X}^c, \mathtt{Y}^c\})$. Finally since cascades are closed under composition, by induction on the comp-height, the proposition follows. Note that ($\star$) is guaranteed by Fact 29.

($2 \Rightarrow 1$) Let $\mathcal{A}$ be a global transducer with input alphabet $\Sigma \times 2^Z$ and output alphabet Σ'. From Lemma 31, we obtain that for every letter $a' \in \Sigma'$, there is a μ-calculus formula over words φ'_a such that on input $w \in (\Sigma \times 2^Z)^*$ and position i, $w, i \models \varphi_{a'}$ iff $a'_1 a'_2 \dots a'_\ell = \mathcal{A}(w)$, $a'_i = a'$. Let $\psi_{a'} \in \mathit{formulas}(\{\mathtt{X}^g, \mathtt{Y}^g\})$ be the formula obtained from φ'_a by replacing $\mathtt{X}$ by $\mathtt{X}^g$, $\mathtt{Y}$ by $\mathtt{Y}^g$, first by fst^g, and last by lst^g. It is clear that on a data word $w \in (\Sigma \times \mathcal{D})^*$ and a position i, it holds that $\mathcal{A}(w) = (a'_1, d_1) \cdots (a'_n, d_n) \in (\Sigma' \times \mathcal{D})^*$ $a'_i = a$ iff $w, i \models \psi_{a'}$. Similarly by replacing $\mathtt{X}$ by $\mathtt{X}^c$, $\mathtt{Y}$ by $\mathtt{Y}^c$, first by fst^c, and last by lst^c, one can obtain a formula $\chi_{a'} \in \mathit{formulas}(\{\mathtt{X}^c, \mathtt{Y}^c\})$ such that for a class transducer $\mathcal{A}$ and for any data word $w \in (\Sigma \times \mathcal{D})^*$ and a position i in w, it is the case that $\mathcal{A}(w) = (a'_1, d_1) \cdots (a'_n, d_n) \in (\Sigma' \times \mathcal{D})^*$ $a'_i = a$ iff $w, i \models \psi_{a'}$. Since BMA is closed under composition by induction on the height of the cascade the claim generalizes to cascades of arbitrary height. Finally the statement holds by observing that these formulas are sufficient to verify that the cascade has a successful run on a given data word. ◀

C Proof of Lemma 13

Proof. Let $\bar{u} = (u_1, \dots, u_n)$ be an n-tuple of words in Σ^* of equal length $2\ell \in \mathbb{N}$ and let $w = w_1 \cdots w_n = \mathit{coding}(\bar{u})$ such that for each i, $\mathit{str}(w_i) = u_i$. Assume that $\mathcal{A}$ has a successful run on w and let $\mathcal{A}(w) = w' = w'_1 \dots w'_n \in (\Gamma \times \mathcal{D})^*$ where w'_i is a relabelling of w_i. Observe that the only other positions in w in the class of a position in w_i appear either in w_{i-1} or w_{i+1} if they exist. Therefore to compute $\mathit{str}(w'_i)$ it suffices to know the words u_{i-1}, u_i, u_{i+1}.

Let $\Sigma' = \Sigma \times 2^Z$. For a word u, we write $u(i)$ for the ith letter of u. Let $f_1 : (\Sigma'^*)^2 \to \Gamma^*, f_2 : (\Sigma'^*)^3 \to \Gamma^*, f_3 : (\Sigma'^*)^2 \to \Gamma^*$ be maps such that for all words $x = x_1 \cdots x_{2m}$,$y = y_1 \cdots y_{2m}$, $z = z_1 \cdots z_{2m}$, $2m \in \mathbb{N}$,

$$f_1(x,y) = \begin{cases} x'_1 \dots x'_{2m} & \text{if } x'_i \neq \epsilon \text{ for all } i \\ \epsilon & \text{otherwise.} \end{cases} \qquad x'_i = \begin{cases} \mathcal{A}(x_i) & \text{if } i \text{ is odd,} \\ \mathcal{A}(x_i y_{i-1})(1) & \text{otherwise.} \end{cases}$$

$$f_2(x,y,z) = \begin{cases} y'_1 \dots y'_{2m} & \text{if } y'_i \neq \epsilon \text{ for all } i \\ \epsilon & \text{otherwise.} \end{cases} \qquad y'_i = \begin{cases} \mathcal{A}(x_{i+1} y_i)(2) & \text{if } i \text{ is odd,} \\ \mathcal{A}(y_i z_{i-1})(1) & \text{otherwise.} \end{cases}$$

$$f_3(y,z) = \begin{cases} z'_1 \dots z'_{2m} & \text{if } z'_i \neq \epsilon \text{ for all } i \\ \epsilon & \text{otherwise.} \end{cases} \qquad z'_i = \begin{cases} \mathcal{A}(z_i) & \text{if } i \text{ is even,} \\ \mathcal{A}(y_{i+1} z_i)(2) & \text{otherwise.} \end{cases}$$

Let $t_1(X,Y), t_2(X,Y,Z), t_3(Y,Z)$, where $X : \Sigma'^*, Y : \Sigma'^*, Z : \Sigma'^*$ be expressions of depth 3 that recognise the maps f_1, f_2, f_3 respectively as guaranteed by Example 9. For the variables $\bar{Y} = Y_1 : \Sigma'^*, \dots, Y_n : \Sigma'^*$, let $s(\bar{Y})$ be the expression

$$s(\bar{Y}) = \mathit{NE}(t_1(Y_1, Y_2), t_2(Y_1, Y_2, Y_3) \dots, t_2(Y_{n-2}, Y_{n-1}, Y_n), t_n(Y_{n-1}, Y_n)) \ .$$

Fix the variables $\bar{X} = X_1 : \Sigma^*, \dots, X_n : \Sigma^*$. Let $\bar{v}(\bar{X})$ stand for the vector of expressions $v_1(X_1) = m_1(X_1), v_2(X_2) = m_2(X_2), \dots, v_{n-1}(X_{n-1}) = m_2(X_{n-1}), v_n(X_n) = m_3(X_n)$. The expressions $e_1(\bar{X}), \dots, e_n(\bar{X})$ are defined as,

$$e_1(\bar{X}) = bI(t_1(v_1(X_1), v_2(X_2)), s(\bar{v}(\bar{X})))$$
$$\text{for each } 1 < i < n, e_i(\bar{X}) = bI(t_2(v_{i-1}(X_{i-1}), v_i(X_i), v_{i+1}(X_{i+1})), s(\bar{v}(\bar{X})))$$
$$e_n(\bar{X}) = bI(t_3(v_{n-1}(X_{n-1}), v_n(X_n)), s(\bar{v}(\bar{X})))$$

The expression $s(\bar{X})$ has depth 6 and hence expressions $e_1(\bar{X}), \dots, e_n(\bar{X})$ has depth 7.

From the definition of the maps f_1, f_2, f_3 and the definition of the coding w it follows that (1) $t_1(v_1(u_1), v_2(u_2)) = \mathit{str}(w'_1)$ (2) for each $1 < i < n$, $t_2(v_{i-1}(u_{i-1}), v_i(u_i), v_{i+1}(u_{i+1})) = \mathit{str}(w'_i)$, and (3) $t_n(v_{n-1}(u_{n-1}), v_n(u_n)) = \mathit{str}(w'_n)$. Hence $s(\bar{v}(\bar{u})) = 1$ and $e_1(\bar{u}) = \mathit{str}(w'_1), \dots, e_n(\bar{u}) = \mathit{str}(w'_n)$. Thus if $\mathcal{A}$ has a successful run on w then $\mathit{coding}(e_1(\bar{u}), \dots, e_n(\bar{u})) = \mathcal{A}(\mathit{coding}(\bar{u}))$.

On the other hand, if $e_1(\bar{u}) = v_1, \dots, e_n(\bar{u}) = v_n$ for identical length words $u_1, \dots, u_n \in \Sigma^*$ then by definition of the maps f_1, f_2, f_3 it follows that the automaton has a successful run on each class of the data word $w = \mathit{coding}(u_1, \dots, u_n)$ and hence $\mathcal{A}(w) = w' = \mathit{coding}(v_1, \dots, v_n)$. ◀

D Proof of Lemma 14

Proof. Assume $\mathcal{A}$ has the set of states Q, set of initial states $I \subseteq Q$ and set of final states $F \subseteq Q$. For $p, q \in Q$, let $\mathcal{A}_{p,q}$ stand for the modified automaton with initial state p and final state q. Let $\bar{u} = (u_1, \dots, u_n)$ be an n-tuple of words in Σ^* of equal length $2\ell \in \mathbb{N}$ and let $w = w_1 \cdots w_n = \mathit{coding}(\bar{u})$ such that for each i, $\mathit{str}(w_i) = u_i$. Assume that $\mathcal{A}$ has a successful run on w and let $\mathcal{A}(w) = w' = w'_1 \dots w'_n \in (\Gamma \times \mathcal{D})^*$ where w'_i is a relabelling of w_i. To compute $\mathit{str}(w'_i)$ it is enough to know the word u_i and the pair $(p, q) \in Q^2$ which is respectively the state of the automaton $\mathcal{A}$ before reading u_i and after reading u_i in the unique run on $\mathit{str}(w)$. Among these, the pair (p, q) can be computed a finitary gate that aggregates the set of all possible partial runs for each of the words $u_1, \dots, u_n$.

We write Σ' for the set $\Sigma \times 2^Z$. Let $f : \Sigma'^* \to \mathcal{P}(Q \times Q)$ be the binary function that maps each word $w \in \Sigma'^*$ to the set of pairs (p, q) such that $\mathcal{A}$ has a run from state p to state q on the word w. Define the finitary functions $l_i : \mathcal{P}(Q \times Q)^i \to \mathcal{P}(Q)$ and $r_i : \mathcal{P}(Q \times Q)^i \to \mathcal{P}(Q)$ as,

$$l((R_1, S_1), \dots, (R_i, S_i)) = \{q \mid (p, q) \in (R_1, S_1) \circ \dots \circ (R_i, S_i), p \in I\}$$
$$r((R_1, S_1), \dots, (R_i, S_i)) = \{p \mid (p, q) \in (R_1, S_1) \circ \dots \circ (R_i, S_i), p \in F\}$$

Further, let $g_1, g_3 : \Sigma'^* \times P(Q) \to \Gamma^*, g_2 : \Sigma'^* \times (P(Q))^2 \to \Gamma^*$ be the binary functions such that for all $u \in \Sigma'^*$ and $Q_1, Q_1 \in \mathcal{P}(Q)$,

$$g_1(u, Q_2) = \begin{cases} \mathcal{A}_{p,q}(u) & \text{if there is a unique } (p,q) \in I \times Q_2 \text{ such that } \mathcal{A}_{p,q} \text{ has a successful run on } u, \\ \epsilon & \text{otherwise.} \end{cases}$$

$$g_2(u, Q_1 Q_2) = \begin{cases} \mathcal{A}_{p,q}(u) & \text{if there is a unique } (p,q) \in Q_1 \times Q_2 \text{ such that } \mathcal{A}_{p,q} \text{ has a successful run on } u, \\ \epsilon & \text{otherwise.} \end{cases}$$

$$g_3(u, Q_1) = \begin{cases} \mathcal{A}_{p,q}(u) & \text{if there is a unique } (p,q) \in Q_1 \times F \text{ such that } \mathcal{A}_{p,q} \text{ has a successful run on } u, \\ \epsilon & \text{otherwise.} \end{cases}$$

Fix the variables $\bar{X} = X_1 : \Sigma^*, \ldots, X_n : \Sigma^*$. Let $\bar{v}(\bar{X})$ stand for the vector of expressions $v_1(X_1) = m_1(X_1), v_2(X_2) = m_2(X_2), \ldots, v_{n-1}(X_{n-1}) = m_2(X_{n-1}), v_n(X_n) = m_3(X_n)$. Define the expressions $e_1(\bar{X}), \ldots, e_n(\bar{X})$ as

$$e_1(\bar{X}) = g_1(v_1(X_1), r(f(v_2(X_2)), \ldots, f(v_n(X_n))))$$

for each$1 < i < n$,

$$e_i(\bar{X}) = g_2(v_i(X_i), cat(l(f(v_1(X_1)), \ldots, f(v_{i-1}(X_{i-1}))), r(f(v_{i+1}(X_{i+1})), \ldots, f(v_n(X_n)))))$$

$$e_n(\bar{X}) = g_3(v_n(X_n), l(f(v_1(X_1)), \ldots, f(v_{n-1}(X_{n-1}))))$$

The expressions $e_1(\bar{X}), \ldots, e_n(\bar{X})$ are of height at most 5.

◀

E Proof of Lemma 15

Proof. Proof by induction on the height of the cascade $\mathcal{C}$. The base case of the induction is when $\mathcal{C}$ has height 1, then $\mathcal{C}$ is either a class transducer or a global transducer and the claim follows by Lemma 13 and Lemma 14. For the inductive step, assume the claim holds for every cascade of height at most $n-1$. Let $\mathcal{C} = \mathcal{C}' \circ \langle \Sigma_1, \mathcal{A}, \Sigma_2 \rangle$ where $\mathcal{C}'$ is a cascade of height $n-1$ and $\mathcal{A}$ a class or global transducer. If $e_1(\bar{X} = X_1, \ldots, X_n), \ldots, e_n(\bar{X})$ and $f_1(\bar{Y} = Y_1, \ldots, Y_n), \ldots, f_n(\bar{Y})$ are expressions given by the induction hypothesis that satisfy the claim for the cascades $\mathcal{A}$ and $\mathcal{C}'$ then the expressions $e_1(f_1(\bar{Y}), \ldots, f_n(\bar{Y})), \ldots, e_n(f_1(\bar{Y}), \ldots, f_n(\bar{Y}))$ satisfy the claim for the cascade $\mathcal{C}$.

◀

F Proof of Proposition 17

Proof. (1 to 2) Let $|u_1| = \ell$. Assume $1 \leq i \leq \ell$ is an index such that the ith letter of each of u_1 to u_n is 1. By the definition of w, the set of positions $\{k \cdot 2\ell + 2i \mid 0 \leq k \leq n-1\}$ in w is labelled by the letter 1. Again by the construction of w, the sequence of positions $2i, 2\ell + 2i - 1, 2\ell + 2i, \ldots, (n-2) \cdot 2\ell + 2i, (n-1) \cdot 2\ell + 2i - 1, (n-1) \cdot 2\ell + 2i$ forms a bridge of length $2n - 1$ that is 1-labelled. Moreover all the positions left of $2i$ or right of $(n-1) \cdot 2\ell + 2i$ belong to singleton classes. Hence the data word w belongs to the language B_n.

(2 to 1) Because of items (1) and (3) of Definition 16, if $w \in B_n$ then there is a bridge that begins on a position $1 \leq 2i \leq 2\ell$. The unique bridge of length of length $2n - 1$ starting from position $2i$ is the sequence of positions $2i, 2\ell + 2i - 1, 2\ell + 2i, \ldots, (n-2) \cdot 2\ell + 2i, (n-1) \cdot 2\ell + 2i - 1, (n-1) \cdot 2\ell + 2i$, which, as we saw already, consists of the ith letters of u_1 to u_n. Hence if the bridge from $2i$ is 1-labelled then the ith letter of u_1 to u_n is 1. ◀