An automaton over data words that captures EMSO logic

Benedikt Bollig

An automaton over data words that captures EMSO logic

Benedikt Bollig

LSV, ENS Cachan, CNRS & INRIA, France
bollig@lsv.ens-cachan.fr

Abstract. We develop a general framework for the specification and implementation of systems whose executions are words, or partial orders, over an infinite alphabet. As a model of an implementation, we introduce class register automata, a one-way automata model over words with multiple data values. Our model combines register automata and class memory automata. It has natural interpretations. In particular, it captures communicating automata with an unbounded number of processes, whose semantics can be described as a set of (dynamic) message sequence charts. On the specification side, we provide a local existential monadic second-order logic that does not impose any restriction on the number of variables. We study the realizability problem and show that every formula from that logic can be effectively, and in elementary time, translated into an equivalent class register automaton.

1 Introduction

A recent research stream, motivated by models from XML database theory, considers *data words*, i.e., strings over an infinite alphabet [2, 10, 14, 21, 23]. The alphabet is the cartesian product of a finite supply of *labels* and an infinite supply of *data values*. While labels may represent, e.g., an XML tag or reveal the type of an action that a system performs, data values can be used to model time stamps [10, 11, 15], process identifiers [6, 25], or text contents in XML documents [4].

We will consider data words as behavioral models of concurrent systems. In this regard, it is natural to look at suitable logics and automata. Logical formulas may serve as specifications, and automata as system models or tools for deciding logical theories. This viewpoint raises the following classical problems/tasks: *satisfiability* (does a given logical formula have a model ?), *model checking* (do all executions of an automaton satisfy a given formula ?), and *realizability* (given a formula, construct a system model in terms of an automaton whose executions are precisely the models of the formula). Much work has indeed gone into defining logics and automata for data words, with a focus on satisfiability [5, 13].

One of the first logical approaches to data words is due to [10]. Since then, a two-variable logic has become a commonly accepted yardstick wrt. expressivity and decidability [5]. The logic contains a predicate to compare data values of two positions for equality. Its satisfiability problem is decidable, indeed, but supposedly of very high complexity. An elementary upper bound has been obtained only for weaker fragments [5, 13]. For specification of communicating systems, however, two-variable logic is of limited use: it cannot express properties like "whenever a process Pid1 spawns some Pid2, then this is followed by a message from Pid2 to Pid1". Actually, the logic was studied for words with only one data value at each each position, which is not enough to encode executions of message-passing systems. But three-variable logics as well as extensions to two data values lead to undecidability. To put it bluntly, any "interesting" logic for dynamic communicating systems has an undecidable satisfiability problem.

Instead of satisfiability or model checking, we therefore consider realizability. A system model that *realizes* a given formula can be considered correct by construction. Realizability questions for data words have, so far, been neglected. One reason may be that there is actually no automaton that could serve as a realistic system model. Though data words naturally reflect executions of systems with an unbounded number of threads, existing automata fail to model distributed computation. Three features are minimum requirements for a suitable system model. First, the automaton should be a *one-way device*, i.e., read an execution once, processing it "from left to right" (unlike data automata [5], class automata [3], two-way register automata, and pebble automata [21]). Second, it should be *non-deterministic* (unlike alternating automata [14,21]). Third, it should reflect paradigms that are used in concurrent programming languages such as process creation and message passing. Two known models match the first two properties: register automata [17,18,25] and class memory automata [2]; but they clearly do not fulfill the last requirement.

Contribution. We provide an existential MSO logic over data words, denoted rEMSO, which does not impose any restriction on the number of variables. The logic is strictly more expressive than the two-variable logic from [5] and suitable to express interesting properties of dynamic communicating systems.

We then define *class register automata* as a system model. They are a mix of register automata [17,18,25] and class memory automata [2]. A class register automaton is a non-deterministic one-way device. Like a class memory automaton, it can access certain configurations in the past. However, we extend the notion of a configuration, which is no longer a simple state but composed of a state *and* some data values that are stored in registers. This is common in concurrent programming languages and can be interpreted as "read current state of a process" or "send process identity from one to another process". Moreover, it is in the spirit of communicating finite-state machines [12] or nested-word automata [1], where more than one resource (state, channel, stack, etc.) can be accessed at a time. Actually, our automata run over directed acyclic graphs rather than words. To our knowledge, they are the first automata model of true concurrency that deals with structures over infinite alphabets.

We study the realizability problem and show that, for every rEMSO formula, we can compute, in elementary time, an equivalent class register automaton. The effective translation is based on Hanf's locality theorem [16] and properly generalizes [7, 9] to a dynamic setting with unbounded process creation.

Outline. Sections 2 and 3 introduce data words and their logics. In Section 4, we define the new automata model. Section 5 is devoted to the realizability problem and states our main result. In Section 6, we give translations from automata back to logic. An extension of our main result to infinite data words is discussed in Section 7. We conclude in Section 8.

2 Data Words

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of natural numbers. For $m \in \mathbb{N}$, we denote by $[m]$ the set $\{1, \ldots, m\}$. A *boolean formula* over a (possibly infinite) set A of *atoms* is a finite object generated by the grammar $\beta ::= \mathit{true} \mid \mathit{false} \mid a \in A \mid \neg\beta \mid \beta \vee \beta \mid \beta \wedge \beta$. For an assignment of truth values to elements of A, a boolean formula β is evaluated to true or false as usual. Its size $|\beta|$ is the number of vertices of its syntax tree. Moreover, $|A| \in \mathbb{N} \cup \{\infty\}$ denotes the size of a set A. The symbol $\cong$ will be used to denote isomorphism of two structures. For a partial function f, the domain of f is denoted by $\mathrm{dom}(f)$.

We fix an infinite set $\mathfrak{D}$ of *data values*. Note that $\mathfrak{D}$ can be *any* infinite set. For examples, however, we usually choose $\mathfrak{D} = \mathbb{N}$. In a data word, every position will carry $m \geq 0$ data values. It will also carry a *label* from a non-empty finite alphabet Σ. Thus, a *data word* is a finite sequence over $\Sigma \times \mathfrak{D}^m$ (over Σ if $m = 0$). Given a data word $w = (a_1, d_1) \ldots (a_n, d_n)$ with $a_i \in \Sigma$ and $d_i = (d_i^1, \ldots, d_i^m) \in \mathfrak{D}^m$, we let $\ell(i)$ refer to label a_i and $d^k(i)$ to data value d_i^k.

Classical words without data come with natural relations on word positions such as the direct successor relation $\prec_{+1}$ and its transitive closure $<$. In the context of data words with one data value (i.e., $m = 1$), it is natural to consider also a relation $\prec_{\sim}$ for successive positions with identical data values [5]. As, in the present paper, we deal with multiple data values, we generalize these notions in terms of a signature. A *signature* $\mathcal{S}$ is a pair $(\sigma, \mathfrak{I})$. It consists of a finite set σ of binary *relation symbols* and an *interpretation* $\mathfrak{I}$. The latter associates, with every $\lhd \in \sigma$ and every data word $w = w_1 \ldots w_n \in (\Sigma \times \mathfrak{D}^m)^*$, a relation $\lhd^w \subseteq [n] \times [n]$ such that the following hold, for all word positions $i, j, i', j' \in [n]$:

(1) $i \lhd^w j$ implies $i < j$
(2) there is at most one k such that $i \lhd^w k$
(3) there is at most one k such that $k \lhd^w i$
(4) if $i \lhd^w j$ and $i' \lhd^w j'$ and $w_i = w_{i'}$ and $w_j = w_{j'}$, then $i < i'$ iff $j < j'$

In other words, we require that $\lhd^w$ (1) complies with $<$, (2) has out-degree at most one, (3) has in-degree at most one, and (4) is monotone. Our translation from logic into automata will be symbolic and independent of $\mathfrak{I}$, but its applicability and correctness rely upon the above conditions. However, several examples will demonstrate that the framework is quite flexible and allows us to capture existing logics and automata for data words. Note that $\lhd^w$ can indeed be *any* relation satisfying (1)–(4). It could even assume an order on $\mathfrak{D}$.

As the interpretation $\mathfrak{I}$ is mostly understood, we may identify $\mathcal{S}$ with σ and write $\lhd \in \mathcal{S}$ instead of $\lhd \in \sigma$, or $|\mathcal{S}|$ to denote $|\sigma|$. If not stated otherwise, we let in the following $\mathcal{S}$ be any signature.

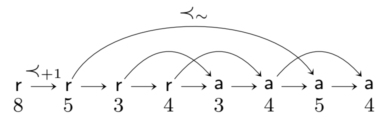

Fig. 1. Data word over $\mathcal{S}^1_{+1,\sim}$

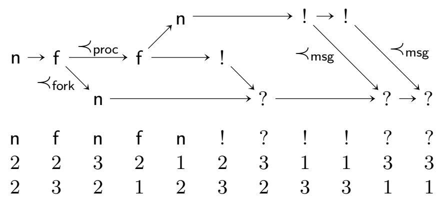

Fig. 2. Data word over $\mathcal{S}^2_{\mathsf{dyn}}$

Example 1. Typical examples of relation symbols include $\prec_{+1}$ and $\prec^k_{\sim}$ relating direct successors and, respectively, successive positions with the same k-th data value: For $w = w_1 \dots w_n$, we let $\prec^w_{+1} = \{(i, i+1) \mid i \in \{1, \dots, n-1\}\}$ and $(\prec^k_{\sim})^w = \{(i,j) \mid 1 \le i < j \le n,\ d^k(i) = d^k(j)$, and there is no $i < i' < j$ such that $d^k(i) = d^k(i')\}$. When $m = 1$, we write $\prec_{\sim}$ instead of $\prec^1_{\sim}$. Automata and logic have been well studied in the presence of one single data value ($m = 1$) and for signature $\mathcal{S}^1_{+1,\sim} = \{\prec_{+1}, \prec_{\sim}\}$ with the above interpretation [2, 5]. Here, and in the following, we adopt the convention that the upper index of a signature denotes the number m of data values. Figure 1 depicts a data word over $\Sigma = \{\mathsf{r}, \mathsf{a}\}$ (request/acknowledgment) and $\mathfrak{D} = \mathbb{N}$ as well as the relations $\prec_{+1}$ (straight arrows) and $\prec_{\sim}$ (curved arrows) imposed by $\mathcal{S}^1_{+1,\sim}$. ◊

Example 2. We develop a framework for message-passing systems with dynamic process creation. Each process has a unique identifier from $\mathfrak{D} = \mathbb{N}$. Process $c \in \mathbb{N}$ can execute an action $\mathsf{f}(c,d)$, which forks a new process with identity d. This action is eventually followed by $\mathsf{n}(d,c)$, indicating that d is new (created by c) and begins its execution. Processes can exchange messages. When c executes $!(c,d)$, it sends a message through an unbounded first-in-first-out (FIFO) channel $c \to d$. Process d may execute $?(d,c)$ to receive the message. Elements from $\Sigma_{\mathsf{dyn}} = \{\mathsf{f}, \mathsf{n}, !, ?\}$ reveal the nature of an action, which requires two identities so that we choose $m = 2$. When a process performs an action, it should access the current state of (i) its own, (ii) the spawning process if a new-action is executed, and (iii) the sending process if a receive is executed (message contents are encoded in states). To this aim, we define a signature $\mathcal{S}^2_{\mathsf{dyn}} = \{\prec_{\mathsf{proc}}, \prec_{\mathsf{fork}}, \prec_{\mathsf{msg}}\}$ with the following interpretation. Assume $w = w_1 \dots w_n \in (\Sigma_{\mathsf{dyn}} \times \mathbb{N} \times \mathbb{N})^*$ and consider, for $a, b \in \Sigma_{\mathsf{dyn}}$ and $i, j \in [n]$, the property

$$P_{(a,b)}(i,j) \;=\; (\ell(i) = a \wedge \ell(j) = b \wedge d^1(i) = d^2(j) \wedge d^2(i) = d^1(j))\,.$$

We set $\prec^w_{\mathsf{proc}} = (\prec^1_{\sim})^w$, which relates successive positions with the same executing process. Moreover, let $i \prec^w_{\mathsf{fork}} j$ if $i < j$, $P_{(\mathsf{f},\mathsf{n})}(i,j)$, and there is no $i < k < j$ such that $P_{(\mathsf{f},\mathsf{n})}(i,k)$ or $P_{(\mathsf{f},\mathsf{n})}(k,j)$. Finally, we set $i \prec^w_{\mathsf{msg}} j$ if $i < j$, $P_{(!,?)}(i,j)$, and

$$|\{i' < i \mid P_{(!,?)}(i',j)\}| \;=\; |\{j' < j \mid P_{(!,?)}(i,j')\}|\,.$$

This models FIFO communication. An example data word is given in Figure 2, which also depicts the relations induced by $\mathcal{S}^2_{\mathsf{dyn}}$. Horizontal arrows reflect $\prec_{\mathsf{proc}}$, vertical arrows either $\prec_{\mathsf{fork}}$ or $\prec_{\mathsf{msg}}$, depending on the labels. Note that $\mathsf{n}(2,2)$ is executed by "root process" 2, which was not spawned by some other process. ◊

Graph Abstraction. Note that the graph induced by the data word from Figure 2 does not resemble a word anymore, as the direct successor relation on word positions is abandoned. Actually, we can see data words from a different angle. A signature $\mathcal{S}$ determines a class of *data graphs* $\mathcal{G}$ with $(\Sigma \times \mathfrak{D}^m)$-labeled nodes and $\mathcal{S}$-labeled edges. A data graph is contained in $\mathcal{G}$ if it can be "squeezed" into a word w such that nodes that are connected by a $\lhd$-labeled edge turn into word positions that are related by $\lhd^w$. In other words, we consider directed acyclic graphs such that at least one linearization (extension to a total order) matches the requirements imposed by the signature.

Our principal proof technique relies on a graph abstraction of data words where data values are classified into equivalence classes. Let $Part(m)$ be the set of all partitions of $[m]$. An $\mathcal{S}$-*graph* is a (node- and edge-labeled) graph $G = (V, (\lhd^G)_{\lhd \in \mathcal{S}}, \lambda, \nu)$. Here, V is the finite set of nodes, $\lambda : V \to \Sigma$ and $\nu : V \to Part(m)$ are node-labeling functions, and each $\lhd^G \subseteq V \times V$ is a set of edges such that, for all $i \in V$, there is at most one $j \in V$ with $i \lhd^G j$, and there is at most one $j \in V$ with $j \lhd^G i$. We represent $\lhd^G$ and $(\lhd^G)^{-1}$ as partial functions and set $\mathsf{next}^G_\lhd(i) = j$ if $i \lhd^G j$, and $\mathsf{prev}^G_\lhd(i) = j$ if $j \lhd^G i$.

Local graph patterns, so-called spheres, will also play a key role. For nodes $i, j \in V$, we denote by $dist^G(i,j)$ the *distance* between i and j, i.e., the length of the shortest path from i to j in the undirected graph $(V\,, \bigcup_{\lhd \in \mathcal{S}} \lhd^G \cup (\lhd^G)^{-1})$ (if such a path exists). In particular, $dist^G(i,i) = 0$. For some *radius* $B \in \mathbb{N}$, the B-*sphere of* G *around* i, denoted by $B\text{-}Sph^G(i)$, is the substructure of G induced by $\{j \in V \mid dist^G(i,j) \leq B\}$. In addition, it contains the distinguished element i as a constant, called *sphere center*.

These notions naturally transfer to data words: With word w of length n, we associate the graph $G(w) = ([n], (\lhd^w)_{\lhd \in \mathcal{S}}, \lambda, \nu)$ where λ maps i to $\ell(i)$ and ν maps i to $\{\{l \in [m] \mid d^k(i) = d^l(i)\} \mid k \in [m]\}$. Thus, $K \in \nu(i)$ contains indices with the same data value at position i. Now, $\mathsf{next}^w_\lhd$, $\mathsf{prev}^w_\lhd$, $dist^w$, and $B\text{-}Sph^w(i)$ are defined with reference to the graph $G(w)$. We hereby assume that $\mathcal{S}$ is understood. We might also omit the index w if it is clear from the context.

Data words u and v are called ($\mathcal{S}$-)*equivalent* if $G(u) \cong G(v)$. For a language L, we let $[L]_\mathcal{S}$ denote the set of words that are equivalent to some word in L.

Given the data word w from Figure 1, we have $dist^w(1,8) = 3$. The picture on the right shows $1\text{-}Sph^w(4)$. The sphere center is framed by a rectangle; node labelings of the form $\{\{1\}\}$ are omitted.

$$\begin{tikzcd}
\mathsf{r} \arrow[r] \arrow[rr, bend left] & \boxed{\mathsf{r}} \arrow[r] \arrow[rr, bend left] & \mathsf{a} \arrow[r] & \mathsf{a}
\end{tikzcd}$$

3 Logic

We consider monadic second-order logic to specify properties of data words. Let us fix countably infinite supplies of first-order variables $\{x, y, \ldots\}$ and second-order variables $\{X, Y, \ldots\}$.

The set MSO($\mathcal{S}$) of *monadic second-order formulas* is given by the grammar

$$\varphi ::= \ell(x) = a \mid d^k(x) = d^l(y) \mid x \lhd y \mid x = y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

where $a \in \Sigma$, $k, l \in [m]$, $\lhd \in \mathcal{S}$, x and y are first-order variables, and X is a second-order variable. The *size* $|\varphi|$ of φ is the number of nodes of its syntax tree.

Important fragments of MSO($\mathcal{S}$) are FO($\mathcal{S}$), the set of first-order formulas, which do not use any second-order quantifier, and EMSO($\mathcal{S}$), the set of formulas of the form $\exists X_1 \ldots \exists X_n \, \varphi$ with $\varphi \in$ FO($\mathcal{S}$).

The models of a formula are data words. First-order variables are interpreted as word positions and second-order variables as sets of positions. Formula $\ell(x) = a$ holds in data word w if position x carries an a, and formula $d^k(x) = d^l(y)$ holds if the k-th data value at position x equals the l-th data value at position y. Moreover, $x \lhd y$ is satisfied if $x \lhd^w y$. The atomic formulas $x = y$ and $x \in X$ as well as quantification and boolean connectives are interpreted as usual.

For realizability, we will actually consider a restricted, more "local" logic: let rMSO($\mathcal{S}$) denote the fragment of MSO($\mathcal{S}$) where we can only use $d^k(x) = d^l(x)$ instead of the more general $d^k(x) = d^l(y)$. Thus, data values of *distinct* positions can only be compared via $x \lhd y$. This implies that rMSO($\mathcal{S}$) cannot distinguish between words u and v such that $G(u) \cong G(v)$. The fragments rFO($\mathcal{S}$) and rEMSO($\mathcal{S}$) of rMSO($\mathcal{S}$) are defined as expected.

In the case of one data value ($m = 1$), we will also refer to the logic EMSO$_2$($\mathcal{S}^1_{+1,\sim} \cup \{<\}$) that was considered in [5] and restricts EMSO logic to two first-order variables. The predicate $<$ is interpreted as the strict linear order on word positions (strictly speaking, it is not part of a signature as we defined it). We shall later see that rEMSO($\mathcal{S}^1_{+1,\sim}$) is strictly more expressive than EMSO$_2$($\mathcal{S}^1_{+1,\sim} \cup \{<\}$), though the latter involves the non-local predicates $d^1(x) = d^1(y)$ and $<$. This gain in expressiveness comes at the price of an undecidable satisfiability problem.

A *sentence* is a formula without free variables. The language defined by sentence φ, i.e., the set of its models, is denoted by $L(\varphi)$. By $\mathbb{MSO}(\mathcal{S})$, r$\mathbb{MSO}(\mathcal{S})$, r$\mathbb{EMSO}(\mathcal{S})$, etc., we refer to the corresponding language classes.

Example 3. Think of a server that can receive requests (r) from an unbounded number of processes, and acknowledge (a) them. We let $\Sigma = \{\mathsf{r}, \mathsf{a}\}$, $\mathfrak{D} = \mathbb{N}$, and $m = 1$. A data value from $\mathfrak{D}$ is used to model the process identity of the requesting and acknowledged process. We present three properties formulated in rFO($\mathcal{S}^1_{+1,\sim}$). Formula $\varphi_1 = \exists x \exists y \, (\ell(x) = \mathsf{r} \wedge \ell(y) = \mathsf{a} \wedge x \prec_\sim y)$ expresses that there is a request that is acknowledged. Dually, $\varphi_2 = \forall x \exists y \, (\ell(x) = \mathsf{r} \rightarrow \ell(y) = \mathsf{a} \wedge x \prec_\sim y)$ says that every request is acknowledged before the same process sends another request. A last formula guarantees that two *successive* requests are acknowledged in the order they were received:

$$\varphi_3 = \forall x, y \left(\begin{array}{c} \ell(x) = \mathsf{r} \wedge \ell(y) = \mathsf{r} \wedge x \prec_{+1} y \\ \rightarrow \exists x', y' \left(\ell(x') = \mathsf{a} \wedge \ell(y') = \mathsf{a} \wedge x \prec_\sim x' \prec_{+1} y' \wedge y \prec_\sim y' \right) \end{array} \right)$$

This is not expressible in EMSO$_2$($\mathcal{S}^1_{+1,\sim} \cup \{<\}$). We will see that $\varphi_1, \varphi_2, \varphi_3$ form a hierarchy of languages that correspond to different automata models, our new model capturing φ_3. $\Diamond$

Example 4. We pursue Example 2 and consider Σ_{dyn} with signature $\mathcal{S}^2_{\mathsf{dyn}}$. Recall that we wish to model systems where an unbounded number of processes communicate via message-passing through unbounded FIFO channels. Obviously, not every data word represents an execution of such a system. Therefore, we identify some *well formed* data words, which have to satisfy $\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \in$ rFO($\mathcal{S}^2_{\mathsf{dyn}}$) given as follows. We require that there is exactly one root process: $\varphi_1 = \exists x \left(\ell(x) = \mathsf{n} \wedge d^1(x) = d^2(x) \wedge \forall y \left(d^1(y) = d^2(y) \rightarrow x = y\right)\right)$. Next, we assume that every fork is followed by a corresponding new-action, the first action of a process is a new-event, and every new process was forked by some other process:

$$\varphi_2 = \forall x \begin{pmatrix} \ell(x) = \mathsf{f} \rightarrow \exists y \, (x \prec_{\mathsf{fork}} y) \\ \wedge \; \ell(x) = \mathsf{n} \leftrightarrow \neg\exists y \, (y \prec_{\mathsf{proc}} x) \\ \wedge \; \ell(x) = \mathsf{n} \rightarrow \left(d^1(x) = d^2(x) \vee \exists y \, (y \prec_{\mathsf{fork}} x)\right) \end{pmatrix}$$

Finally, every send should be followed by a receive, and a receive be preceded by a send action: $\varphi_3 = \forall x \left(\ell(x) \in \{\,!\,,?\,\} \rightarrow \exists y \, (x \prec_{\mathsf{msg}} y \vee y \prec_{\mathsf{msg}} x)\right)$. This formula actually ensures that, for every $c, d \in \mathbb{N}$, there are as many symbols $!(c,d)$ as $?(d,c)$, the N-th send symbol being matched with the N-th receive symbol. We call a data word over Σ_{dyn} and $\mathcal{S}^2_{\mathsf{dyn}}$ a *message sequence chart* (MSC, for short) if it satisfies $\varphi_1 \wedge \varphi_2 \wedge \varphi_3$. Figure 2 shows an MSC and the induced relations. When we restrict to MSCs, our logic corresponds to that from [20]. Note that model checking rMSO($\mathcal{S}^2_{\mathsf{dyn}}$) specifications against *fork-and-join grammars*, which can generate infinite sets of MSCs, is decidable [20].

A last rFO($\mathcal{S}^2_{\mathsf{dyn}}$)-formula (which is not satisfied by all MSCs) specifies that, whenever a process c forks some d, then this is followed by a message from d to c: $\forall x_1, y_1 \; (x_1 \prec_{\mathsf{fork}} y_1 \rightarrow \exists x_2, y_2 \, (x_1 \prec_{\mathsf{proc}} x_2 \wedge y_1 \prec_{\mathsf{proc}} y_2 \prec_{\mathsf{msg}} x_2))$. $\Diamond$

4 Class Register Automata

In this section, we define class register automata, a non-deterministic one-way automata model that captures rEMSO logic. It combines register automata [17,18] and class memory automata [2]. When processing a data word, data values from the current position can be stored in registers. The automaton reads the data word from left to right but can look back on certain states and register contents from the past (e.g., at the last position that is executed by the same process). Positions that can be accessed in this way are determined by the signature $\mathcal{S}$. Their register entries can be compared with one another, or with current values from the input. Moreover, when taking a transition, registers can be updated by either a current value, an old register entry, or a guessed value.

Definition 1. *A* class register automaton *(over signature $\mathcal{S}$) is a tuple $\mathcal{A} = (Q, R, \Delta, (F_{\lhd})_{\lhd \in \mathcal{S}}, \Phi)$ where*

- *Q is a finite set of* states*,*
- *R is a finite set of* registers*,*

– *the* $F_\lhd \subseteq Q$ *are sets of* local final states*,*
– Φ *is the* global acceptance condition: *a boolean formula over* $\{$ '$q \leq N$' $\mid q \in Q$ *and* $N \in \mathbb{N}\}$*, and*
– Δ *is a finite set of* transitions *of the form*

$$(p, g) \xrightarrow{a} (q, f) \,.$$

Here, $p : \mathcal{S} \rightharpoonup Q$ *is a partial mapping representing the source states. Moreover,* g *is a guard, i.e., a boolean formula over* $\{$ '$\theta_1 = \theta_2$' $\mid \theta_1, \theta_2 \in [m] \cup (\mathrm{dom}(p) \times R)\}$ *to perform comparisons of values that are are currently read and those that are stored in registers. Finally,* $a \in \Sigma$ *is the current label,* $q \in Q$ *is the target state, and* $f : R \rightharpoonup (\mathrm{dom}(p) \times R) \cup ([m] \times \mathbb{N})$ *is a partial mapping to update registers.*

In the following, we write $p_\lhd$ instead of $p(\lhd)$. Transition $(p, g) \xrightarrow{a} (q, f)$ can be executed at position i of a data word if the state at position $\mathsf{prev}_\lhd(i)$ is $p_\lhd$ (for all $\lhd \in \mathrm{dom}(p)$) and, for a register guard $(\lhd_1, r_1) = (\lhd_2, r_2)$, the entry of register r_1 at $\mathsf{prev}_{\lhd_1}(i)$ equals that of r_2 at $\mathsf{prev}_{\lhd_2}(i)$. The automaton then reads the label a together with a tuple of data values that also passes the test given by g, and goes to q. Moreover, register r obtains a new value according to $f(r)$: if $f(r) = (\lhd, r') \in \mathrm{dom}(p) \times R$, then the new value of r is the value of r' at position $\mathsf{prev}_\lhd(i)$; if $f(r) = (k, B) \in [m] \times \mathbb{N}$, then r obtains any k-th data value in the B-sphere around i. In particular, $f(r) = (k, 0)$ assigns to r the (unique) k-th data value of the current position. To some extent, $f(r) = (k, B)$ calls an oracle to guess a data value. The guess is local and, therefore, weaker than [18], where a non-deterministic reassignment allows one to write *any* data value into a register. This latter approach can indeed simulate our local version (this is not immediately clear, but can be shown using the *sphere automaton* from Section 5).

Let us be more precise. A configuration of $\mathcal{A}$ is a pair (q, ρ) where $q \in Q$ is the current state and $\rho : R \rightharpoonup \mathfrak{D}$ is a partial mapping denoting the current register contents. If $\rho(r)$ is undefined, then there is no entry in r. Let $w = w_1 \dots w_n \in (\Sigma \times \mathfrak{D}^m)^*$ be a data word and $\xi = (q_1, \rho_1) \dots (q_n, \rho_n)$ be a sequence of configurations. For $i \in [n]$, $k \in [m]$, and $B \in \mathbb{N}$, let $\mathfrak{D}^k_B(i) = \{d^k(j) \mid j \in [n]$ such that $\mathit{dist}^w(i, j) \leq B\}$. We call ξ a *run* of $\mathcal{A}$ on w if, for every position $i \in [n]$, there is a transition $(p_i, g_i) \xrightarrow{\ell(i)} (q_i, f_i)$ such that the following hold:

(1) $\mathrm{dom}(p_i) = \{\lhd \in \mathcal{S} \mid \mathsf{prev}_\lhd(i)$ is defined$\}$

(2) for all $\lhd \in \mathrm{dom}(p_i) : (p_i)_\lhd = q_{\mathsf{prev}_\lhd(i)}$

(3) g_i is evaluated to true on the basis of its atomic subformulas: $\theta_1 = \theta_2$ is true iff $\mathit{val}_i(\theta_1) = \mathit{val}_i(\theta_2) \in \mathfrak{D}$ where $\mathit{val}_i(k) = d^k(i)$ and $\mathit{val}_i((\lhd, r)) = \rho_{\mathsf{prev}_\lhd(i)}(r)$ (the latter might be undefined and, therefore, not be in $\mathfrak{D}$)

(4) for all $r \in R$: $\begin{cases} \rho_i(r) = \rho_{\mathsf{prev}_\lhd(i)}(r') & \text{if } f_i(r) = (\lhd, r') \in \mathrm{dom}(p) \times R \\ \rho_i(r) \in \mathfrak{D}^k_B(i) & \text{if } f_i(r) = (k, B) \in [m] \times \mathbb{N} \\ \rho_i(r) \text{ undefined} & \text{if } f_i(r) \text{ undefined} \end{cases}$

Run ξ is accepting if $q_i \in F_\lhd$ for all $i \in [n]$ and $\lhd \in \mathcal{S}$ such that $\mathsf{next}_\lhd(i)$ is undefined. Moreover, we require that the global condition Φ is met. Hereby, an atomic constraint $q \leq N$ is satisfied by ξ if $|\{\, i \in [n] \mid q_i = q\}| \leq N$. The language $L(\mathcal{A}) \subseteq (\Sigma \times \mathfrak{D}^m)^*$ of $\mathcal{A}$ is defined in the obvious manner. The corresponding language class is denoted by $\mathbb{CRA}(\mathcal{S})$.

The acceptance conditions are inspired by Björklund and Schwentick [2], who also distinguish between local and global acceptance. Local final states can be motivated as follows. When data values model process identities, a $\prec_\sim$-maximal position of a data word is the last position of some process and must give rise to a local final state. Moreover, in the context of $\mathcal{S}^2_{\mathsf{dyn}}$, a sending position that does not lead to a local final state in $F_{\prec_{\mathsf{msg}}}$ requires a matching receive event. Thus, local final states can be used to model "communication requests". The global acceptance condition of class register automata is more general than that of [2] to cope with all possible signatures. However, in the special case of $\mathcal{S}^1_{+1,\sim}$, there is some global control in terms of $\prec_{+1}$. We could then perform some counting up to a finite threshold and restrict, like [2], to a set of global final states.

We can classify many of the non-deterministic one-way models from the literature (most of them defined for $m = 1$) in our unifying framework:

- A *class memory automaton* [2] is a class register automaton where, in all transitions $(p, g) \xrightarrow{a} (q, f)$, the update function f is undefined everywhere. The corresponding language class is denoted by $\mathbb{CMA}(\mathcal{S})$.
- As an intermediary subclass of class register automata, we consider *non-guessing class register automata*: for all transitions $(p, g) \xrightarrow{a} (q, f)$ and registers r, one requires $f(r) \in (\mathrm{dom}(p) \times R) \cup ([m] \times \{0\})$. We denote the corresponding language class by $\mathbb{CRA}^-(\mathcal{S})$.
- A *register automaton* [14,17] is a non-guessing class register automaton over $\mathcal{S}^m_{+1} = \{\prec_{+1}\}$. Moreover, non-guessing class register automata over $\mathcal{S}^1_{+1,\sim}$ capture *fresh-register automata* [25], which can dynamically generate data values that do not occur in the history of a run. Actually, this feature is also present in dynamic communicating automata [6] and in class memory automata over $\mathcal{S}^1_{+1,\sim}$ where a fresh data value is guaranteed by a transition $(p, g) \xrightarrow{a} (q, f)$ such that $p_{\prec_\sim}$ is undefined.
- Class register automata are a model of distributed computation: considered over Σ_{dyn} and $\mathcal{S}^2_{\mathsf{dyn}}$, they subsume dynamic communicating automata [6]. In particular, they can handle unbounded process creation and message passing. Updates of the form $f(r) = (\prec_{\mathsf{fork}}, r')$ and $f(r) = (\prec_{\mathsf{msg}}, r')$ correspond to receiving a process identity from the spawning/sending process. Moreover, when a process requests a message from the thread whose identity is stored in register r, a corresponding transition is guarded by $(\prec_{\mathsf{proc}}, r) = (\prec_{\mathsf{msg}}, r_0)$ where we assume that every process keeps its identity in some register r_0.

Example 5. Let us give a concrete example. Suppose $\Sigma = \{\mathsf{r}, \mathsf{a}\}$ and $\mathfrak{D} = \mathbb{N}$. We pursue Example 3 and build a non-guessing class register automaton $\mathcal{A}$ over $\mathcal{S}^1_{+1,\sim}$ for $L = [\{(\mathsf{r}, 1) \ldots (\mathsf{r}, n)(\mathsf{a}, 1) \ldots (\mathsf{a}, n) \mid n \geq 1\}]_{\mathcal{S}^1_{+1,\sim}}$. Roughly speaking,

Transitions

	source (p)		guard (g)	input	q	update (f)
	$\prec_{\sim}$	$\prec_{+1}$				
1				(r, d)	q_1	$r_1 := d$
2		q_1		(r, d)	q_1	$r_1 := d$ $r_2 := (\prec_{+1}, r_1)$
3	q_1	q_1	$(\prec_{\sim}, r_2) = \perp$	(a, d)	q_2	$r_1 := d$
4	q_1	q_2	$(\prec_{\sim}, r_2) = (\prec_{+1}, r_1)$	(a, d)	q_2	$r_1 := d$

Run

input	state	r_1	r_2
$(\mathsf{r}, 8)$	q_1	8	$\perp$
$(\mathsf{r}, 5)$	q_1	5	8
$(\mathsf{a}, 8)$	q_2	8	$\perp$
$(\mathsf{a}, 5)$	q_2	5	$\perp$

Fig. 3. A non-guessing class register automaton over $\mathcal{S}^1_{+1,\sim}$ and a run

there is a request phase followed by an acknowledgment phase, and requests are acknowledged in the order they are received. Figure 3 presents $\mathcal{A}$ and an accepting run on $(\mathsf{r}, 8)(\mathsf{r}, 5)(\mathsf{a}, 8)(\mathsf{a}, 5)$. The states of $\mathcal{A}$ are q_1 and q_2. State q_1 is assigned to request positions (first phase), state q_2 to acknowledgments (second phase). Moreover, $\mathcal{A}$ is equipped with registers r_1 and r_2. During the first phase, r_1 always contains the data value of the current position, and r_2 the data value of the $\prec_{+1}$-predecessor (unless we deal with the very first position, where r_2 is undefined, denoted $\perp$). These invariants are ensured by transitions 1 and 2. In the second phase, by transition 3, position $n+1$ carries the same data value as the first position, which is the only request with undefined r_2. Guard $(\prec_{\sim}, r_2) = \perp$ is actually an abbreviation for $\neg((\prec_{\sim}, r_2) = (\prec_{\sim}, r_2))$. By transition 4, position $n + i$ with $i \geq 2$ has to match the request position whose r_2-contents equals r_1 at $n + i - 1$. Finally, $F_{\prec_{\sim}} = \{q_2\}$, $F_{\prec_{+1}} = \{q_2\}$, and $\Phi = \neg(q_1 \leq 0)$. ♢

For the language L from Example 5, one can show $L \notin \mathbb{CMA}(\mathcal{S}^1_{+1,\sim})$, using an easy pumping argument. Next, we will see that non-guessing class register automata, though more expressive than class memory automata, are not yet enough to capture rEMSO logic. Thus, dropping just one feature such as registers or guessing data values makes class register automata incomparable to the logic. Assume $m = 2$ and consider $\mathcal{S}^2_{\sim} = \{\prec^1_{\sim}, \prec^2_{\sim}\}$ (cf. Example 1).

Lemma 1. $\mathrm{r}\mathbb{FO}(\mathcal{S}^2_{\sim}) \not\subseteq \mathbb{CRA}^{-}(\mathcal{S}^2_{\sim})$.

Proof. We determine a formula $\varphi \in \mathrm{rFO}(\mathcal{S}^2_{\sim})$ and show, by contradiction, that every non-guessing class register automaton capturing $L = L(\varphi)$ will necessarily accept a data word outside L. Roughly speaking, L consists of words where every position belongs to a pattern that is depicted in Figure 4 and captured by the formula $\mathit{pattern}(x_1, \ldots, x_4) = x_1 \prec^1_{\sim} x_3 \wedge x_1 \prec^2_{\sim} x_4 \wedge x_2 \prec^2_{\sim} x_3 \wedge x_2 \prec^1_{\sim} x_4$. With this, $\varphi = \forall x \exists x_1, \ldots, x_4\, (x \in \{x_1, \ldots, x_4\} \wedge \mathit{pattern}(x_1, \ldots, x_4)) \in \mathrm{rFO}(\mathcal{S}^2_{\sim})$ is the formula for L. Suppose that there is a non-guessing class register automaton $\mathcal{A}$ over $\mathcal{S}^2_{\sim}$ recognizing L. We build a data word $w = (a, d_1) \ldots (a, d_n) \in L$ with $n \in 4\mathbb{N}$ and $\{d^1_1, \ldots, d^1_n\} \cap \{d^2_1, \ldots, d^2_n\} = \emptyset$ by nesting disjoint patterns as depicted in Figure 4: we first create $i_1, \ldots, i_4$, then add $j_1, \ldots, j_4$; the next pattern is to be inserted at $n_1, \ldots, n_4$, etc. We assume that the data values of distinct patterns are disjoint. If we choose n large enough, then there are an accepting run $\xi = (q_1, \rho_1) \ldots (q_n, \rho_n)$ of $\mathcal{A}$ on w (with transition $t_i = (p_i, g_i) \xrightarrow{a} (q_i, f_i)$ at

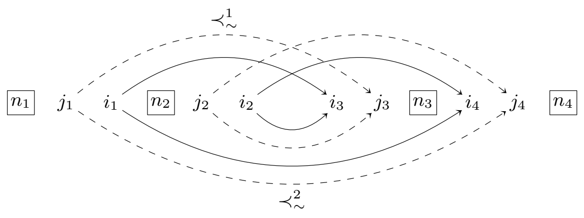

Fig. 4. Nested patterns

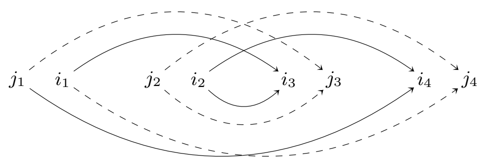

Fig. 5. Merging patterns

position i) and positions $i_1, \ldots, i_4, j_1, \ldots, j_4$ of w such that $j_1 < i_1$, $i_1, \ldots, i_4$ and $j_1, \ldots, j_4$ form two (disjoint) patterns, and $t_{i_1} = t_{j_1}, \ldots, t_{i_4} = t_{j_4}$. Now, consider the data word w' that we obtain from w when we swap the second data values of positions i_1 and j_1. Thus, the data part of w' is

$$d_1 \ldots d_{j_1-1}\,(d^1_{j_1}, d^2_{i_1})\,d_{j_1+1}\ \ldots\ d_{i_1-1}\,(d^1_{i_1}, d^2_{j_1})\,d_{i_1+1} \ldots d_n\,.$$

We have the situation depicted in Figure 5. In particular, $i_1, \ldots, i_4$ do not form a single closed cycle. This violates φ, as $x = i_1$ implies $x_1 \in \{i_1, i_2\}$. Thus, $w' \notin L$. However, applying transitions $t_1, \ldots, t_n$ still yields an accepting run $\xi' = (q_1, \rho'_1) \ldots (q_n, \rho'_n)$ of $\mathcal{A}$ on w'. For $i \in [n]$, ρ'_i is then given as follows:

$$\rho'_i(r) = \begin{cases} d^2_{i_1} & \text{if } \rho_i(r) = d^2_{j_1} \text{ and } i \in \{j_1, j_3\} \\ d^2_{j_1} & \text{if } \rho_i(r) = d^2_{i_1} \text{ and } i \in \{i_1, i_3\} \\ d^1_{i_1} & \text{if } \rho_i(r) = d^1_{j_1} \text{ and } i = j_4 \\ d^1_{j_1} & \text{if } \rho_i(r) = d^1_{i_1} \text{ and } i = i_4 \\ \rho_i(r) & \text{otherwise} \end{cases}$$

One can verify that ξ' is indeed an accepting run on w'. $\square$

The proof of Lemma 1 can be adapted to show $\mathrm{r}\mathbb{FO}(\mathcal{S}^2_{\mathsf{dyn}}) \not\subseteq \mathbb{CRA}^-(\mathcal{S}^2_{\mathsf{dyn}})$. It reveals that non-guessing class register automata can in general not detect *cycles*. However, this is needed to capture rFO logic [16]. In Section 5, we show that *full* class register automata capture rFO and, as they are closed under projection, also rEMSO logic. Closure under projection is meant in the following sense. Let Γ be a non-empty finite alphabet. Given $\mathcal{S} = (\sigma, \mathfrak{I})$, we define another signature $\mathcal{S}_\Gamma$ for data words over $(\Sigma \times \Gamma) \times \mathfrak{D}^m$. Its set of relation symbols is $\{\lhd_\Gamma \mid \lhd \in \mathcal{S}\}$. For $w \in ((\Sigma \times \Gamma) \times \mathfrak{D}^m)^*$, we set $i \lhd^w_\Gamma j$ iff $i \lhd^{proj_\Sigma(w)} j$. Hereby, the projection

$proj_{\Sigma}$ just removes the Γ component while keeping Σ and the data values. For $\mathcal{C} \in \{\mathbb{CRA}, \mathbb{CRA}^{-}, \mathbb{CMA}\}$, we say that $\mathcal{C}(\mathcal{S})$ is *closed under projection* if, for every Γ and $L \subseteq ((\Sigma \times \Gamma) \times \mathfrak{D}^m)^*$, $L \in \mathcal{C}(\mathcal{S}_{\Gamma})$ implies $proj_{\Sigma}(L) \in \mathcal{C}(\mathcal{S})$.

Lemma 2. *For every signature $\mathcal{S}$, $\mathbb{CRA}(\mathcal{S})$, $\mathbb{CRA}^{-}(\mathcal{S})$, and $\mathbb{CMA}(\mathcal{S})$ are closed under union, intersection, and projection. They are, in general, not closed under complementation.*

Proof. Closure under union and intersection follows standard automata-theoretic constructions. Closure under projection holds since projection preserves the graph structure of a data word. For non-complementability, we can rely on the corresponding result for communicating automata [9]. Roughly speaking, a communicating automaton is a dynamic communicating automaton with a fixed set of at least two processes *Proc*. It can be identified as a special case of our framework: We let $m = 0$, since the number of processes is fixed. Moreover, $\Sigma = \{\, !(c,d)\,, ?(c,d) \mid c,d \in \mathit{Proc}$ such that $c \neq d \,\}$ is the set of actions. Finally, we define the signature $\mathcal{S}^0_{\mathit{Proc}} = \{\prec_{\mathsf{proc}}\,, \prec_{\mathsf{msg}}\}$ as the straightforward restriction of $\mathcal{S}^2_{\mathsf{dyn}}$ (cf. Example 2) to this bounded case. Speaking in terms of our framework, [9] indeed shows that class register automata (or, as $m = 0$, class memory automata) over $\mathcal{S}^0_{\mathit{Proc}}$ are not closed under complementation. □

5 Realizability of EMSO Specifications

In this section, we solve the realizability problem for rEMSO specifications:

Theorem 1. *For all signatures $\mathcal{S}$, $\mathrm{r}\mathbb{EMSO}(\mathcal{S}) \subseteq \mathbb{CRA}(\mathcal{S})$. An automaton can be computed in elementary time and is of elementary size.*

Classical procedures that translate formulas into automata follow an inductive approach, use two-way mechanisms and tools such as pebbles, or rely on reductions to existing translations. There is no obvious way to apply any of these techniques to prove our theorem.

We therefore follow a technique from [9], which is based on ideas from [22,24]. We first transform the first-order kernel of the formula at hand into a normal form due to Hanf [16]. According to that normal form, satisfaction of a first-order formula wrt. data word w only depends on the spheres that occur in $G(w)$, and on how often they occur, counted up to a threshold. The size of a sphere is bounded by a radius that depends on the formula. The threshold can be computed from the radius and $|\mathcal{S}|$. We can indeed apply Hanf's Theorem, as the structures that we consider have *bounded degree*: every node/word position has at most $|\mathcal{S}|$ incoming and at most $|\mathcal{S}|$ outgoing edges. In a second step, we transform the formula in normal form into a class register automaton.

Recall that $B\text{-}Sph^{G}(i)$ denotes the B-sphere of graph/data word G around i (cf. Section 2). Its size (number of nodes) is bounded by $maxSize := (2|\mathcal{S}| + 2)^B$. Let $B\text{-}Spheres_{\mathcal{S}} = \{B\text{-}Sph^{G}(i) \mid G = (V, \ldots)$ is an $\mathcal{S}$-graph and $i \in V\}$. We do not distinguish between isomorphic structures so that $B\text{-}Spheres_{\mathcal{S}}$ is finite.

Theorem 2 (cf. [8, 16]). *Let $\varphi \in \mathrm{rFO}(\mathcal{S})$. One can compute, in elementary time, $B \in \mathbb{N}$ and a boolean formula β over $\{$'$S \leq N$' $\mid S \in B\text{-}Spheres_{\mathcal{S}}$ and $N \in \mathbb{N}\}$ such that $L(\varphi)$ is the set of data words that satisfy β. Here, we say that $w = w_1 \dots w_n$ satisfies atom $S \leq N$ iff $|\{i \in [n] \mid B\text{-}Sph^w(i) \cong S\}| \leq N$. The radius B and the size of β and its constants N are elementary in $|\varphi|$ and $|\mathcal{S}|$.*

Proof. A simple but crucial observation is that there exists a first-order sentence that is equivalent to φ but talks about $G(w)$ rather than w. We simply write $\lambda(x) = a$ instead of $\ell(x) = a$, and $\bigvee_{\eta \in \mathcal{P}} \nu(x) = \eta$ instead of $d^k(x) = d^l(x)$ where $\mathcal{P} \subseteq Part(m)$ is the set of partitions of $[m]$ such that k and l occur in the same set. As rMSO($\mathcal{S}$)-formulas cannot distinguish between data words that induce the same graph, the boolean formula β in normal form exists due to [16]. Actually, β can be computed in triply exponential time [8]. $\square$

By Theorem 2, it will be useful to have a class register automaton that, when reading a position i of data word w, outputs the sphere of w around i. Its construction is actually the main difficulty in the proof of Theorem 1, as spheres have to be computed "in one go", i.e., reading the word from left to right, while accessing only certain configurations from the past.

Proposition 1. *Let $B \in \mathbb{N}$. One can compute, in elementary time, a class register automaton $\mathcal{A}_B = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, true)$ over $\mathcal{S}$, as well as a mapping $\pi : Q \to B\text{-}Spheres_{\mathcal{S}}$ such that $L(\mathcal{A}_B) = (\Sigma \times \mathfrak{D}^m)^*$ and, for every data word $w = w_1 \dots w_n$, every accepting run $(q_1, \rho_1) \dots (q_n, \rho_n)$ of $\mathcal{A}_B$ on w, and every $i \in [n]$, $\pi(q_i) \cong B\text{-}Sph^w(i)$. Moreover, $|Q|$ and $|R|$ are elementary in B and $|\mathcal{S}|$.*

The proposition is proved below. Let us first show how we can use it, together with Theorem 2, to translate an rEMSO formula into a class register automaton.

Proof (of Theorem 1). Let $\varphi = \exists X_1 \dots \exists X_n\, \psi \in \mathrm{rEMSO}(\mathcal{S})$ be a sentence with $\psi \in \mathrm{rFO}(\mathcal{S})$ (we also assume $n \geq 1$). Since Theorem 2 applies to first-order formulas only, we extend Σ to $\Sigma \times \Gamma$ where $\Gamma = 2^{\{1,\dots,n\}}$. Consider the extended signature $\mathcal{S}_\Gamma$ (cf. Section 4). From ψ, we obtain a formula $\psi_\Gamma \in \mathrm{rFO}(\mathcal{S}_\Gamma)$ by replacing $\ell(x) = a$ with $\bigvee_{M \in \Gamma} \ell(x) = (a, M)$ and $x \in X_j$ with $\bigvee_{a \in \Sigma,\, M \in \Gamma} \ell(x) = (a, M \cup \{j\})$. Consider the radius $B \in \mathbb{N}$ and the normal form β_Γ for ψ_Γ due to Theorem 2. Let $\mathcal{A}_B = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}_\Gamma}, true)$ be the class register automaton over $\mathcal{S}_\Gamma$ from Proposition 1 and π be the associated mapping. The global acceptance condition of $\mathcal{A}_B$ is obtained from β_Γ by replacing every atom $S \leq N$ with $\pi^{-1}(S) \leq N$ (which can be expressed as a suitable boolean formula). We hold $\mathcal{A}'_B$, a class register automaton satisfying $L(\mathcal{A}'_B) = L(\psi_\Gamma)$. Exploiting closure under projection (Lemma 2), we obtain a class register automaton over $\mathcal{S}$ that recognizes $L(\varphi) = proj_\Sigma(L(\psi_\Gamma))$. $\square$

The Sphere Automaton. In the remainder of this section, we construct the class register automaton $\mathcal{A}_B = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, true)$ from Proposition 1, together with $\pi : Q \to B\text{-}Spheres_{\mathcal{S}}$. The idea is that, at each position i in the data word w at hand, $\mathcal{A}_B$ guesses the B-sphere S of w around i. To verify that the guess is correct, i.e., $S \cong B\text{-}Sph^w(i)$, S is passed to each position that is connected to i by an edge in $G(w)$. That new position locally checks label and data equalities imposed by S, then also forwards S to its neighbors, and so on. Thus, at any time, several local patterns have to be validated simultaneously so that a state $q \in Q$ is actually a *set* of spheres. In fact, we consider *extended* spheres $E = (S, \alpha, col)$ where $S = (U, (\lhd^E)_{\lhd \in \mathbb{S}}, \lambda, \nu, \gamma)$ is a sphere (with universe U and sphere center γ), $\alpha \in U$ is the *active node*, and col is a color from a finite set, which will be specified later. The active node α indicates the current context, i.e., it corresponds to the position currently read.

Let $B\text{-}eSpheres_{\mathbb{S}}$ denote the set of extended spheres, which is finite up to isomorphism. For $E = (S, \alpha, col) \in B\text{-}eSpheres_{\mathbb{S}}$, $S = (U, (\lhd^E)_{\lhd \in \mathbb{S}}, \lambda, \nu, \gamma)$, and $j \in U$, we let $E[j]$ refer to the extended sphere (S, j, col) where the active node α has been replaced with j. Now suppose that the state q of $\mathcal{A}_B$ that is reached after reading position i of data word w contains $E = (S, \alpha, col)$. Roughly speaking, this means that the neighborhood of i in w shall look like the neighborhood of α in S. Thus, if S contains j' such that $\alpha \lhd^E j'$, then we must find i' such that $i \lhd^w i'$ in the data word. Local final states will guarantee that i' indeed exists. Moreover, the state assigned to i' in a run of $\mathcal{A}_B$ will contain the new proof obligation $E[j']$ and so forth. Similarly, an edge in (the graph of) w has to be present in spheres, unless it is beyond their scope, which is limited by B. All this is reflected below, in conditions T2–T6 of a transition.

We are still facing two major difficulties. Several *isomorphic* spheres have to be verified simultaneously, i.e., a state must be allowed to include isomorphic spheres in different contexts. A solution to this problem is provided by the additional coloring col. It makes sure that centers of overlapping isomorphic spheres with different colors refer to distinct nodes in the input word. To put it differently, for a given position i in data word w, there may be i' such that $0 < dist^w(i, i') \leq 2B+1$ and $B\text{-}Sph^w(i) \cong B\text{-}Sph^w(i')$. Fortunately, there cannot be more than $(2|\mathbb{S}|+1) \cdot maxSize^2$ such positions. As a consequence, the coloring col can be restricted to the set $\{1, \ldots, (2|\mathbb{S}| + 1) \cdot maxSize^2 + 1\}$.

Implementing these ideas alone would do without registers and yield a class memory automaton. But this cannot work due to Lemma 1. Indeed, a faithful simulation of cycles in spheres has to make use of data values. They need to be anticipated, stored in registers, and locally compared with current data values from the input word. We introduce a register (E, k) for every extended sphere E and $k \in [m]$. To get the idea behind this, consider a run $(q_1, \rho_1) \ldots (q_n, \rho_n)$ of $\mathcal{A}_B$ on $w = (a_1, d_1) \ldots (a_n, d_n)$. Pick a position i of w and suppose that $E = (U, (\lhd^E)_{\lhd \in \mathbb{S}}, \lambda, \nu, \gamma, \alpha, col) \in q_i$. If α is minimal in E, then there is no pending requirement to check. Now, as α shall correspond to the current position i of w, we write, for every $k \in [m]$, d_i^k into register (E, k) (first case of T8 below). For all $j \in U \setminus \{\alpha\}$, on the other hand, we anticipate data values and store them in $(E[j], k)$ (also first case of T8). They will be forwarded (second case of T8) and checked later against both the guesses made at other minimal nodes of E (guard g_3 of T7) and the actual data values in w (guard g_2). This procedure makes sure that the values that we carry along within an accepting run agree with the actual data values of w.

Now, as $\mathsf{prev}^w_\lhd$ and $\mathsf{next}^w_\lhd$ are monotone wrt. positions with identical labels and data values, two isomorphic cycles cannot be "merged" into one larger one, unlike in non-guessing class register automata where different parts may act erroneously on the assumption of inconsistent data values (cf. Lemma 1). As a consequence, spheres are correctly simulated by the input word.

Let us formalize $\mathcal{A}_B = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, \mathit{true})$ and the mapping $\pi : Q \to B\text{-}\mathit{Spheres}_{\mathcal{S}}$, following the above ideas. The set of registers is $R = B\text{-}\mathit{eSpheres}_{\mathcal{S}} \times [m]$. A state from Q is a non-empty set $q \subseteq B\text{-}\mathit{eSpheres}_{\mathcal{S}}$ such that

(i) there is a unique $E = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma, \alpha, \mathit{col}) \in q$ such that $\gamma = \alpha$ (we set $\pi(q) = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$ to obtain the mapping required by Prop. 1),

(ii) there are $a \in \Sigma$ and $\eta \in \mathit{Part}(m)$ such that, for all $E = (\ldots, \lambda, \nu, \ldots) \in q$, we have $\lambda(\alpha) = a$ and $\nu(\alpha) = \eta$ (we let $\mathit{label}(q) = a$ and $\mathit{data}(q) = \eta$), and

(iii) for every $(S, \alpha, \mathit{col}), (S, \alpha', \mathit{col}) \in q$, we have $\alpha = \alpha'$.

Before we turn to the transitions, we introduce some notation. Below, E will always denote $(S, \alpha, \mathit{col})$ with $S = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$; in particular, α refers to the active node of E. The mappings $\mathsf{next}^E_\lhd$, $\mathsf{prev}^E_\lhd$, and dist^E are defined for extended spheres in the obvious manner. For $j \in U$, we set $\mathit{type}^-(j) = \{\lhd \in \mathcal{S} \mid \mathsf{prev}^E_\lhd(j) \text{ is defined}\}$. Let us fix, for all $E \in B\text{-}\mathit{eSpheres}_{\mathcal{S}}$ such that $\mathit{type}^-(\alpha) \neq \emptyset$, some arbitrary $\lhd_E \in \mathit{type}^-(\alpha)$. Finally, for state q and $k_1, k_2 \in [m]$, we write $k_1 \sim_q k_2$ if there is $K \in \mathit{data}(q)$ such that $\{k_1, k_2\} \subseteq K$.

We have a transition $(p, g) \xrightarrow{a} (q, f)$ iff the following hold:

T1 $\mathit{label}(q) = a$

T2 for all $\lhd \in \mathcal{S}$, $E \in q$: $\quad \lhd \notin \mathrm{dom}(p) \implies \mathsf{prev}^E_\lhd(\alpha)$ is undefined

T3 for all $\lhd \in \mathrm{dom}(p)$, $E \in q$, $j \in U$: $\quad j \lhd^E \alpha \iff E[j] \in p_\lhd$

T4 for all $\lhd \in \mathrm{dom}(p)$, $E \in p_\lhd$, $j \in U$: $\quad \alpha \lhd^E j \iff E[j] \in q$

T5 for all $\lhd \in \mathrm{dom}(p)$, $E \in q$: $\quad \mathsf{prev}^E_\lhd(\alpha)$ undefined $\implies \mathit{dist}^E(\gamma, \alpha) = B$

T6 for all $\lhd \in \mathrm{dom}(p)$, $E \in p_\lhd$: $\quad \mathsf{next}^E_\lhd(\alpha)$ undefined $\implies \mathit{dist}^E(\gamma, \alpha) = B$

T7 $g = g_1 \wedge g_2 \wedge g_3$ where

$$g_1 = \bigwedge_{\substack{k_1, k_2 \in [m] \\ k_1 \sim_q k_2}} k_1 = k_2 \;\wedge \bigwedge_{\substack{k_1, k_2 \in [m] \\ k_1 \not\sim_q k_2}} \neg\, (k_1 = k_2) \qquad g_2 = \bigwedge_{\substack{k \in [m] \;\; E \in q \\ \lhd \in \mathit{type}^-(\alpha)}} k = (\lhd, (E, k))$$

$$g_3 = \bigwedge_{\substack{k \in [m] \quad E \in q \quad j \in U \\ \lhd_1, \lhd_2 \in \mathit{type}^-(\alpha)}} (\lhd_1, (E[j], k)) = (\lhd_2, (E[j], k))$$

T8 for all $k \in [m]$ and $E \in B\text{-}\mathit{eSpheres}_{\mathcal{S}}$:

$$f((E, k)) = \begin{cases} (k, \mathit{dist}^E(j, \alpha)) & \text{if } \exists j \in U : E[j] \in q \text{ and } \mathit{type}^-(j) = \emptyset \\ (\lhd_{E[j]}, (E, k)) & \text{if } \exists j \in U : E[j] \in q \text{ and } \mathit{type}^-(j) \neq \emptyset \\ \text{undefined} & \text{otherwise} \end{cases}$$

For every $\lhd \in \mathcal{S}$, the local acceptance condition is given by $F_\lhd = \{q \in Q \mid$ for all $E \in q$, $\mathsf{next}^E_\lhd(\alpha)$ is undefined$\}$. Recall that the global one is *true*.

As the maximal size of a sphere is exponential in B and polynomial in $|\mathcal{S}|$, the numbers $|Q|$ and $|R|$ are elementary in B and $|\mathcal{S}|$. Note that $\mathcal{A}_B$ can actually be constructed in elementary time.

In the appendix, we show that the construction of $\mathcal{A}_B$ and π is correct in the sense of Proposition 1.

6 From Automata to Logic

Next, we give translations from automata back to logic. Note that $\mathrm{r}\mathbb{EMSO}(\mathcal{S}^1_{+1}) \subsetneq \mathbb{CRA}(\mathcal{S}^1_{+1})$, as $\mathrm{rEMSO}(\mathcal{S}^1_{+1})$ cannot reason about data values. However, we show that the behavior of a class register automaton is always MSO definable and, in a sense, "regular". There are natural finite-state automata that do not share this property: two-way register automata (even deterministic ones) over one-dimensional data words are incomparable to $\mathrm{MSO}(\mathcal{S}^1_{+1,\sim})$ [21].

Theorem 3. *For every signature $\mathcal{S}$, we have $\mathbb{CRA}(\mathcal{S}) \subseteq \mathbb{MSO}(\mathcal{S})$.*

Proof. As usual, second-order variables are used to encode an assignment of positions to transitions, which is then checked for being an accepting run. To simulate register contents, we extend a technique from [21]. Let us describe how a class register automaton $\mathcal{A} = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, \Phi)$ over $\mathcal{S}$ is translated into an $\mathrm{MSO}(\mathcal{S})$-sentence $\varphi_\mathcal{A}$ such that $L(\varphi_\mathcal{A}) = L(\mathcal{A})$. Suppose $\mathcal{B}$ is the maximum of all B for which there is a transition $(p, g) \xrightarrow{a} (q, f) \in \Delta$ with $f(r) = (k, B)$, for some r and k.

We assume a second-order variable X_δ for every transition $\delta \in \Delta$. Moreover, we assume a variable $X^\beta_{r,B}$ for each $r \in R$, $B \in \{1, \ldots, \mathcal{B}\}$, and each formula $\beta(x_\mathrm{u}, x_\mathrm{v}) \in \mathrm{rFO}(\mathcal{S})$, with free variables x_u and x_v, that is of the form

$$\beta(x_\mathrm{u}, x_\mathrm{v}) = \exists x_1, \ldots, x_B \, (x_\mathrm{u} \bowtie_1 x_1 \bowtie_2 \ldots \bowtie_B x_B = x_\mathrm{v})$$

where $\bowtie_i \in \{=, \lhd, \lhd^{-1} \mid \lhd \in \mathcal{S}\}$. The intuition of these variables is as follows. If a position x is contained in X_δ with $\delta = (p, g) \xrightarrow{a} (q, f)$ and $f(r) = (k, B)$, then x will also be contained in some $X^\beta_{r,B}$, meaning that x executes δ and the new data value of r is the k-th data value at the unique y such that $\beta(x, y)$ is satisfied.

The formula $\varphi_\mathcal{A}$ will be of the form $\exists (X_\delta)_\delta \, \exists (X^\beta_{r,B})_{r,B,\beta} \, (\psi_1 \wedge \psi_2)$. Here, $\psi_1 \in \mathrm{rFO}(\mathcal{S})$ checks whether the following hold:

- each position x is contained in exactly one set X_δ
- for all x and $r \in R$, x is contained in at most one set of the form $X^\beta_{r,B}$
- if $x \in X_\delta$ with $\delta = (p, g) \xrightarrow{a} (q, f)$ and $f(r) = (k, B)$, then $x \in X^\beta_{r,B}$ for some β
- the label at position $x \in X_\delta$ corresponds to the label of δ
- conditions (1) and (2) in the definition of a run are met

$$\begin{tikzcd}
\mathbb{MSO}(\mathcal{S}^1_{+1,\sim}) \arrow[r, phantom, "="] & \mathrm{r}\mathbb{MSO}(\mathcal{S}^1_{+1,\sim}) \\
\mathbb{EMSO}(\mathcal{S}^1_{+1,\sim}) \arrow[u, dashed] & \mathbb{CRA}(\mathcal{S}^1_{+1,\sim}) \arrow[u] \\
\mathrm{r}\mathbb{EMSO}(\mathcal{S}^1_{+1,\sim}) \arrow[u, dashed] \arrow[ur, dashed, "\text{Thm. 1}"'] & \mathbb{CRA}^-(\mathcal{S}^1_{+1,\sim}) \arrow[u, dashed] \\
\mathbb{EMSO}_2(\mathcal{S}^1_{+1,\sim} \cup \{<\}) \arrow[u] \arrow[r, phantom, "="] & \mathbb{CMA}(\mathcal{S}^1_{+1,\sim}) \arrow[u] \arrow[r, phantom, "{[2,5]}"] & {} \\
\mathbb{CRA}(\mathcal{S}^1_{+1}) \arrow[u, "{[2]}"'] \arrow[r, phantom, "="] & \mathbb{CRA}^-(\mathcal{S}^1_{+1})
\end{tikzcd}$$

Fig. 6. A hierarchy of automata and logics over one-dimensional data words

– the (potential) run is accepting, i.e., $F_\lhd$ and Φ are respected

It remains to define $\psi_2 \in \mathrm{MSO}(\mathcal{S})$ to check property (3) of a run. This can be done by means of formulas $\psi_g(x)$, one for each atomic guard $g \in \{\, \theta_1 = \theta_2 \mid \theta_1, \theta_2 \in [m] \cup (\mathcal{S} \times R)\}$. We restrict here to $g = ((\lhd, r) = l)$ with $l \in [m]$. The other cases are similar. Formula $\psi_g(x)$ checks if the contents of r at position $\mathsf{prev}_\lhd(x)$ equals the l-th data value at x. It will be of the form $\exists X\, \exists (X_r)_{r \in R}\, \chi_g$. The idea is that the positions in X describe a path $x_1 \lhd_1 x_2 \lhd_2 \ldots \lhd_{n-1} x_n \lhd x$ that "transports" the data value $d^l(x)$. We suppose that every position x_i is contained in precisely one set X_{r_i} meaning that register r_i is updated by the contents of r_{i-1} at position x_{i-1}. More precisely, we require that, for all $i \in \{2, \ldots, n\}$, there is a transition δ with register-update mapping f such that $x_i \in X_\delta$ and $f(r_i) = (\lhd_{i-1}, r_{i-1})$. The last update should concern r, i.e., we require $x_n \in X_r$. So suppose $x_1 \in X_{r_1}$. It remains to ensure that register r_1, at x_1, obtains the value $d^l(x)$. More precisely, there should be a transition δ with update mapping f, as well as k, B, β and a position x_0 such that $\beta(x_1, x_0)$ holds, $f(r_1) = (k, B)$, $x_1 \in X_\delta \cap X^\beta_{r_1,B}$ and $d^k(x_0) = d^l(x)$.

Note that χ_g can be defined as an FO($\mathcal{S}$)-formula and $\psi_g(x)$ holds iff the register contents of r at $\mathsf{prev}_\lhd(x)$ equals $d^l(x)$. $\square$

In the proof, the non-local predicate $d^k(x) = d^l(y)$ is indeed essential to simulate register assignments, as we need to compare data values at positions where registers are updated. For one-dimensional data words, however, the predicate can be easily defined in rMSO($\mathcal{S}^1_{+1,\sim}$). The following theorem is dedicated to this classical setting over $\mathcal{S}^1_{+1,\sim}$.

Theorem 4. *We have the inclusions depicted in Figure 6. Here,* $\longrightarrow$ *means 'strictly included' and* $\dashrightarrow$ *means 'included'.*

Proof. The inclusion r$\mathbb{EMSO}(\mathcal{S}^1_{+1,\sim}) \subseteq \mathbb{CRA}(\mathcal{S}^1_{+1,\sim})$ is due to Theorem 1, and $\mathbb{CRA}(\mathcal{S}^1_{+1,\sim}) \subseteq \mathbb{MSO}(\mathcal{S}^1_{+1,\sim})$ is due to Theorem 3. The equality $\mathbb{MSO}(\mathcal{S}^1_{+1,\sim}) =$ r$\mathbb{MSO}(\mathcal{S}^1_{+1,\sim})$ is obvious.

$\mathbb{CMA}(\mathcal{S}^1_{+1,\sim}) \subsetneqq \mathbb{rEMSO}(\mathcal{S}^1_{+1,\sim})$: Consider a class memory automaton $\mathcal{A}$. As $\mathcal{A}$ is completely state-based and does not make use of any register, it is standard to define a sentence $\psi \in \text{rEMSO}(\mathcal{S}^1_{+1,\sim})$ such that $L(\psi) = L(\mathcal{A})$. It remains to show strictness of the inclusion.[1] Suppose $\Sigma = \{\mathsf{r}, \mathsf{a}\}$ and $\mathfrak{D} = \mathbb{N}$, and let $L = [\{(\mathsf{r}, 1) \ldots (\mathsf{r}, n)(\mathsf{a}, 1) \ldots (\mathsf{a}, n) \mid n \geq 1\}]_{\mathcal{S}^1_{+1,\sim}}$ (note that the proof also works if Σ is a singleton). Towards a contradiction, suppose L is recognized by class memory automaton $\mathcal{A}$. As $\mathcal{A}$ has no access to registers, a run of $\mathcal{A}$ on $(\mathsf{r}, 1) \ldots (\mathsf{r}, n)(\mathsf{a}, 1) \ldots (\mathsf{a}, n)$ is actually a sequence of states $q_1 \ldots q_{2n}$. If n is large enough, there are positions $1 \leq i < j \leq n$ such that $q_i = q_j$. Now, we can simply exchange the data values at positions i and j without affecting acceptance. More precisely, $q_1 \ldots q_{2n}$ is also an accepting run on the data word $(\mathsf{r}, 1) \ldots (\mathsf{r}, i{-}1)(\mathsf{r}, j)(\mathsf{r}, i{+}1) \ldots (\mathsf{r}, j{-}1)(\mathsf{r}, i)(\mathsf{r}, j{+}1) \ldots (\mathsf{r}, n)(\mathsf{a}, 1) \ldots (\mathsf{a}, n)$, which is not contained in L, a contradiction. On the other hand, L is the conjunction $\varphi_1 \wedge \varphi_2$ of the following $\text{rFO}(\mathcal{S}^1_{+1,\sim})$-sentences:

- $\varphi_1 = \exists x\, \mathit{true} \wedge \forall x\, \exists^{=1} y \begin{pmatrix} x \prec_\sim y \wedge \ell(x) = \mathsf{r} \wedge \ell(y) = \mathsf{a} \\ \vee\, y \prec_\sim x \wedge \ell(y) = \mathsf{r} \wedge \ell(x) = \mathsf{a} \end{pmatrix}$
- $\varphi_2 = \forall x, y \begin{pmatrix} x \prec_{+1} y \wedge \neg(\ell(x) = \mathsf{r} \wedge \ell(y) = \mathsf{a}) \\ \to \exists x', y' \begin{pmatrix} x \prec_\sim x' \prec_{+1} y' \wedge y \prec_\sim y' \\ \vee\, x' \prec_{+1} y' \prec_\sim y \wedge x' \prec_\sim x \end{pmatrix} \end{pmatrix}$

The first formula expresses that the word has positive length and each $\sim$ equivalence class has size two. The second formula ensures the FIFO structure of a data word.

$\mathbb{CMA}(\mathcal{S}^1_{+1,\sim}) \subsetneqq \mathbb{CRA}^-(\mathcal{S}^1_{+1,\sim})$: Consider the language L from the previous paragraph. It is not in $\mathbb{CMA}(\mathcal{S}^1_{+1,\sim})$. However, Example 5 demonstrates that there is a non-guessing class register automaton recognizing L.

$\mathbb{rMSO}(\mathcal{S}^1_{+1,\sim}) \not\subseteq \mathbb{CRA}(\mathcal{S}^1_{+1,\sim})$: We encode grids into data words. An (i, j)-*grid* is a graph that has a height $i \in \mathbb{N}$ and a width $j \in \mathbb{N}$ meaning that it has i rows and j columns that are connected by a horizontal and a vertical immediate successor relation. Nodes are labeled by elements from $\Sigma = \{a, b, c\}$. We encode an (i, j)-grid as the data word

$$(a_{11}, 1) \ldots (a_{i1}, i)(a_{12}, 1) \ldots (a_{i2}, i) \ldots\ldots (a_{1j}, 1) \ldots (a_{ij}, i)$$

where $a_{kl} \in \Sigma$ is the labeling of the grid node (k, l). Hereby, each subword $(a_{1k}, 1) \ldots (a_{ik}, i)$ constitutes a column. Then, moving down in the grid corresponds to a $\prec_{+1}$-step in the data word, moving right corresponds to a $\prec_\sim$-step. These steps are $\text{rFO}(\mathcal{S}^1_{+1,\sim})$-definable.

Consider the set $\mathcal{L}$ of grids of the form $H_1.C.H_2$ where C is a single column of c-labeled nodes, and H_1 and H_2 are grids with labels from $\{a, b\}$ such that the sets of different column words (over $\{a, b\}$) in H_1 and H_2 coincide. We know

[1] Note that satisfiability of $\text{rEMSO}(\mathcal{S}^1_{+1,\sim})$ is undecidable, whereas emptiness of class memory automata over $\mathcal{S}^1_{+1,\sim}$ is decidable [5]. This already implies that there is no *effective* translation of automata into formulas.

that $\mathcal{L}$ is MSO-definable in the signature of a grid. Therefore, the encoding L of $\mathcal{L}$ into data words is rMSO($\mathbb{S}^1_{+1,\sim}$)-definable. Using an argument from [24], we show that $L \notin \mathbb{CRA}(\mathbb{S}^1_{+1,\sim})$. First observe that the number of distinct sets of columns words over $\{a, b\}$ of length n is 2^{2^n}. Suppose, towards a contradiction, that there is a class register automaton $\mathcal{A} = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathbb{S}}, \Phi)$ such that $L(\mathcal{A}) = L$. Without loss of generality, we assume that Φ is given in terms of a simple set of global final states. In a run of $\mathcal{A}$ on the data-word encoding of grid $H_1.C.H_2$ of height n, all the information that $\mathcal{A}$ has about H_1 must be encoded in the n configurations that are taken while reading the c-labeled positions. The number of tuples of n configurations that $\mathcal{A}$ can distinguish is bounded by

$$N = |Q|^n \cdot 2^{(|R| \cdot n)^2} \cdot (n+1)^{|R| \cdot n} .$$

Here, the second factor is an upper bound on the number of equivalence classes on the set $\{1, \ldots, |R| \cdot n\}$, which captures guessed values, and the third factor is the number of registers assignments. Now, as Q and R are fixed, N does not grow sufficiently fast so that $\mathcal{A}$ will accept a data word outside L.

$\mathbb{CRA}(\mathbb{S}^1_{+1}) \subseteq \mathbb{CRA}^-(\mathbb{S}^1_{+1})$: Note first that class register automata over $\mathbb{S}^1_{+1}$ are a variant of the *register automata with non-deterministic reassignment* from [18]. The crucial difference is that the "look-ahead" of $\mathbb{CRA}(\mathbb{S}^1_{+1})$ is bounded, while the automata from [18] can guess any arbitrary data value. As a consequence, the latter capture the set of data words such that all data values (except the last one) are different from the last data value. We will show that, on the other hand, class register automata over $\mathbb{S}^1_{+1}$ are no more expressive than classical register automata, which cannot recognize that language.

Let $\mathcal{A} = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathbb{S}}, \Phi)$ be a class register automaton over $\mathbb{S}^1_{+1}$. We sketch the construction of a non-guessing class register automaton $\mathcal{A}' = (Q', R', \Delta', (F'_\lhd)_{\lhd \in \mathbb{S}}, \Phi')$ over $\mathbb{S}^1_{+1}$ such that $L(\mathcal{A}) = L(\mathcal{A}')$. Let $\mathcal{B}$ be the maximal value B such that an update of $\mathcal{A}$ is of the form $f(r) = (k, B)$. Without loss of generality, we assume that $B \geq 1$ exists. The idea is that $\mathcal{A}'$ keeps track of the register contents of the last B positions, and of the last B data values read. To this aim, we set $R' = \{-\mathcal{B}, \ldots, -1\} \times (R \uplus \{\mathsf{current}\})$. Register $(-i, \mathsf{current})$ contains the i-th last input data value (wrt. the next position to read), and register $(-i, r)$ simulates register assignments of $\mathcal{A}$ for r. In particular, this allows us to access every input data value from the last $B \leq \mathcal{B}$ positions. In order to anticipate data values, a state of $\mathcal{A}'$ contains, apart from a state of $\mathcal{A}$, an equivalence relation over both the new set of registers R' and the next B positions. Thus, a state of $\mathcal{A}'$ is a pair $(q, \sim)$ where $q \in Q$ and $\sim$ is an equivalence relation over $R' \times \{1, \ldots, \mathcal{B}\}$.

To simulate an update $f(r) = (k, B)$ of $\mathcal{A}$ with $B \geq 1$, $\mathcal{A}'$ either writes the current value or one of the values stored in $(-B, \mathsf{current}), \ldots, (-1, \mathsf{current})$ into r, or goes into a state in which r and at least one of the next B positions are considered equivalent. Of course, the equivalence has to be *globally consistent* and *locally consistent* meaning that two equivalent registers should contain the same data value. Moreover, when $\mathcal{A}'$ is in a state where the next position and a defined register r are considered equivalent, then the next symbol to read is the contents of r. If, in contrast, the next position is not equivalent to some defined register, then $\mathcal{A}'$ should read a data value that is currently not stored, and store it in r (unless another update for r applies). This finally ensures that a suitable data value in terms of an equivalence relation has been guessed when performing an update of the form $f(r) = (k, B)$. $\square$

The remaining (strict) inclusions are left open. When there are no data values, we have expressive equivalence of EMSO logic and class register automata (which then reduce to class memory automata). The translation from automata to logic follows the standard approach. The following theorem is a proper generalization of the main result of [9].

Theorem 5. *Suppose $m = 0$. For every signature $\mathcal{S}$, $\mathbb{EMSO}(\mathcal{S}) = \mathbb{CRA}(\mathcal{S})$.*

7 Infinite Data Words

In the realm of reactive systems, it is appropriate to consider infinite data words, i.e., sequences from the set $(\Sigma \times \mathfrak{D}^m)^\omega$. Note that all the notions that we introduced in Section 2 carry over to the new domain. In particular, a formula from rMSO($\mathcal{S}$) is interpreted over an infinite word w without modifying the definition. However, its fragment rEMSO($\mathcal{S}$) now appears limited. In terms of $\mathcal{S}^2_{\mathsf{dyn}}$, one cannot express "some process sends infinitely many messages during an execution", as can be shown using Hanf's Theorem. We therefore introduce a first-order quantifier $\exists^\infty$. Formula $\exists^\infty x\, \varphi$ is satisfied by $w = w_1 w_2 \ldots \in (\Sigma \times \mathfrak{D}^m)^\omega$ if there are infinitely many positions $i \geq 1$ such that φ is satisfied when x is interpreted as i. We obtain the logics rFO$^\infty$($\mathcal{S}$) and rEMSO$^\infty$($\mathcal{S}$) as well as the language class r$\mathbb{EMSO}^\infty$($\mathcal{S}$). Now, a translation from logic into automata requires an extension of class register automata. We define an *ω-class register automaton* (over $\mathcal{S}$) to be a tuple $\mathcal{A} = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, \Phi)$ where $Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}$ are as in class register automata, and Φ is henceforth a boolean formula over $\{$ '$q = \infty$' $\mid q \in Q\} \cup \{$ '$q \leq N$' $\mid q \in Q$ and $N \in \mathbb{N}\}$. Infnite runs $(q_1, \rho_1)(q_2, \rho_2) \ldots$ and satisfaction of the new global acceptance condition are defined as one would expect. In particular, atom $q = \infty$ is satisfied if $|\{i \geq 1 \mid q_i = q\}| = \infty$. The class of languages recognized by ω-class register automata is denoted by ω-$\mathbb{CRA}(\mathcal{S})$. Theorems 1 and 5 extend to infinite words.

Theorem 6. *For all $\mathcal{S}$, we have r$\mathbb{EMSO}^\infty(\mathcal{S}) \subseteq \omega$-$\mathbb{CRA}(\mathcal{S})$. The size of the automaton is elementary in the size of the formula and $|\mathcal{S}|$. If $m = 0$, then r$\mathbb{EMSO}^\infty(\mathcal{S}) = \omega$-$\mathbb{CRA}(\mathcal{S})$.*

Proof. The crucial observation is that Proposition 1 still holds. We actually take the same automaton $\mathcal{A}_B$ and run it on infinite words. The argument that makes the construction work relies on the fact that the *past* of any word position is finite. Moreover, it was shown in [7] that Theorem 2 has a counterpart for formulas with infinity quantifier. The proof is based on Vinner's extension of Ehrenfeucht-Fraïssé games [26]. Thus, for $\varphi \in$ rFO$^\infty$($\mathcal{S}$), there are $B \in \mathbb{N}$ and a boolean formula β over $\{$'$S = \infty$', '$S \leq N$' $\mid S \in B\text{-}Spheres_{\mathcal{S}}$ and $N \in \mathbb{N}\}$ such that $L(\varphi)$ is the set of data words that satisfy β. With this, the constructions from Section 5 can be adapted to translate an rEMSO$^{\infty}(\mathcal{S})$-sentence into an ω-class register automaton over $\mathcal{S}$. $\square$

We remark that the proof of Theorem 6 is not effective. Unlike the proof of Theorem 1, it does not rely on [8, 16]. We do not know if there is an effective alternative.

8 Conclusion

We studied the realizability problem for data-word languages. A particular case of this general framework constitutes a first step towards a logically motivated automata theory for dynamic message-passing systems. In light of this, it would be desirable to synthesize smaller and deadlock-free automata from logical or algebraic specifications. A good starting point for those studies may be temporal logic [14, 19].

Our approach to modeling systems over infinite alphabets may also lead to meaningful model-checking questions. It would be interesting to extend [20], whose logic corresponds to ours in the case of $\mathcal{S}^2_{\mathsf{dyn}}$, to general data words.

References

1. R. Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56(3):1–43, 2009.
2. H. Björklund and Th. Schwentick. On notions of regularity for data languages. *Theoretical Computer Science*, 411(4-5):702–715, 2010.
3. M. Bojańczyk and S. Lasota. An extension of data automata that captures XPath. In *LICS 2010*, pages 243–252. IEEE Computer Society, 2010.
4. M. Bojańczyk, A. Muscholl, Th. Schwentick, and L. Segoufin. Two-variable logic on data trees and applications to XML reasoning. *Journal of the ACM*, 56(3), 2009.
5. M. Bojańczyk, A. Muscholl, Th. Schwentick, L. Segoufin, and C. David. Two-variable logic on words with data. In *LICS 2006*, pages 7–16. IEEE Computer Society, 2006.
6. B. Bollig and L. Hélouët. Realizability of dynamic MSC languages. In F.M. Ablayev and E.W. Mayr, editors, *CSR 2010*, volume 6072 of *LNCS*, pages 48–59, 2010.
7. B. Bollig and D. Kuske. Muller message-passing automata and logics. *Information and Computation*, 206(9-10):1084–1094, 2008.
8. B. Bollig and D. Kuske. An optimal construction of Hanf sentences, 2011. arXiv:1105.5487.
9. B. Bollig and M. Leucker. Message-passing automata are expressively equivalent to EMSO logic. *Theoretical Computer Science*, 358(2):150–172, 2006.
10. P. Bouyer. A logical characterization of data languages. *Information Processing Letters*, 84(2):75–85, 2002.
11. P. Bouyer, A. Petit, and D. Thérien. An algebraic approach to data languages and timed languages. *Information and Computation*, 182(2):137–162, 2003.

12. D. Brand and P. Zafiropulo. On communicating finite-state machines. *Journal of the ACM*, 30(2), 1983.
13. C. David, L. Libkin, and T. Tan. On the satisfiability of two-variable logic over data words. In C.G. Fermüller and A. Voronkov, editors, *LPAR 2010*, LNCS, pages 248–262. Springer, 2010.
14. S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. *ACM Transactions on Computational Logic*, 10(3), 2009.
15. D. Figueira, P. Hofman, and S. Lasota. Relating timed and register automata. In S.B. Fröschle and F.D. Valencia, editors, *EXPRESS'10*, 2010.
16. W. Hanf. Model-theoretic methods in the study of elementary logic. In J. W. Addison, L. Henkin, and A. Tarski, editors, *The Theory of Models*. North-Holland, Amsterdam, 1965.
17. M. Kaminski and N. Francez. Finite-memory automata. *Theoretical Computer Science*, 134(2):329–363, 1994.
18. M. Kaminski and D. Zeitlin. Finite-memory automata with non-deterministic reassignment. *International Journal of Foundations of Computer Science*, 21(5):741–760, 2010.
19. A. Kara, Th. Schwentick, and Th. Zeume. Temporal logics on words with multiple data values. In K. Lodaya and M. Mahajan, editors, *FSTTCS 2010*, volume 8 of *LIPIcs*, pages 481–492, 2010.
20. M. Leucker, P. Madhusudan, and S. Mukhopadhyay. Dynamic message sequence charts. In M. Agrawal and A. Seth, editors, *FSTTCS 2002*, volume 2556 of *LNCS*, pages 253–264. Springer, 2002.
21. F. Neven, Th. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Transactions on Computational Logic*, 5(3):403–435, 2004.
22. Th. Schwentick and K. Barthelmann. Local normal forms for first-order logic with applications to games and automata. *Discrete Mathematics & Theoretical Computer Science*, 3(3):109–124, 1999.
23. L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In Z. Ésik, editor, *CSL 2006*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.
24. W. Thomas. Elements of an automata theory over partial orders. In *POMIV 1996*, volume 29 of *DIMACS*. AMS, 1996.
25. N. Tzevelekos. Fresh-register automata. In Th. Ball and M. Sagiv, editors, *POPL 2011*, pages 295–306. ACM, 2011.
26. S. Vinner. A generalization of Ehrenfeucht's game and some applications. *Israel Journal of Mathematics*, 12:279–286, 1972.

A. Correctness of sphere automaton

We will show that the class register automaton $\mathcal{A}_B = (Q, R, \Delta, (F_\lhd)_{\lhd \in \mathcal{S}}, \Phi)$ over $\mathcal{S}$ and the mapping $\pi : Q \to B\text{-}\mathit{Spheres}_{\mathcal{S}}$ are correct in the sense of Proposition 1: $L(\mathcal{A}_B) = (\Sigma \times \mathfrak{D}^m)^*$ and, for every data word $w = w_1 \dots w_n$ (where $w_i = (a_i, d_i)$), every accepting run $(q_1, \rho_1) \dots (q_n, \rho_n)$ of $\mathcal{A}_B$ on w, and every position $i \in [n]$, $\pi(q_i) \cong B\text{-}\mathit{Sph}^w(i)$.

Every data word is accepted. Let us first show $L(\mathcal{A}_B) = (\Sigma \times \mathfrak{D}^m)^*$, i.e., that every data word is accepted by $\mathcal{A}_B$. Let $w = (a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times \mathfrak{D}^m)^*$ be any data word and let $G(w) = ([n], (\lhd^w)_{\lhd \in \mathcal{S}}, \hat{\lambda}, \hat{\nu})$ be its associated graph. We have to show $w \in L(\mathcal{A}_B)$. A key issue is the assignment of colors to word positions in w such that overlapping spheres can be verified simultaneously. Let $i, i' \in [n]$. We say that i and i' have a *B-overlap* in w if both $B\text{-}\mathit{Sph}^w(i) \cong B\text{-}\mathit{Sph}^w(i')$ and $\mathit{dist}^w(i, i') \leq 2B + 1$.

Lemma 3. *There is a mapping $\Phi : [n] \to \{1, \dots, (2|\mathcal{S}| + 1) \cdot \mathit{maxSize}^2 + 1\}$ such that $\Phi(i) \neq \Phi(i')$ whenever i and i' are distinct and have a B-overlap.*

Proof. We obtain Φ as a coloring of the undirected graph $([n], \mathit{Arcs})$ where two nodes are connected iff they are distinct and have a B-overlap. The graph has degree at most $(2|\mathcal{S}| + 1) \cdot \mathit{maxSize}^2$ so that it can be $((2|\mathcal{S}| + 1) \cdot \mathit{maxSize}^2 + 1)$-colored by some mapping Φ, i.e., $\Phi(i) \neq \Phi(i')$ for every edge $\{i, i'\}$. $\square$

We now define a sequence $\xi = (q_1, \rho_1) \dots (q_n, \rho_n)$ of configurations of $\mathcal{A}_B$ and show that ξ is an accepting run of $\mathcal{A}_B$ on w. Let $i \in [n]$. We set

$$q_i = \{\, (B\text{-}\mathit{Sph}^w(i_c), i, \Phi(i_c)) \;\mid\; i_c \in [n] \text{ such that } \mathit{dist}^w(i_c, i) \leq B \,\}.$$

Suppose $E = (S, \alpha, \mathit{col})$, $S = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$, and $k \in [m]$. We define $\rho_i((E, k))$ as follows. If there are positions $i_c, i' \in [n]$ such that $\mathit{dist}^w(i_c, i) \leq B$, $\mathit{dist}^w(i_c, i') \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_c), i')$, and $\mathit{col} = \Phi(i_c)$, then we set $\rho_i((E, k)) = d^k(i')$. Otherwise, we let $\rho_i((E, k))$ be undefined. Note that $\rho_i((E, k))$ is well defined, as there is at most one pair i_c, i' satisfying the above properties.

We check that q_i is a state. Let $E = (S, \alpha, \mathit{col}) \in q_i$ and $E' = (E', \alpha', \mathit{col}') \in q_i$ with $S = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$ and $S' = (U', (\lhd^{E'})_{\lhd \in \mathcal{S}}, \lambda', \nu', \gamma')$.

(i) Assume $\gamma = \alpha$ and $\gamma' = \alpha'$. Then, $(S, \gamma) \cong (B\text{-}\mathit{Sph}^w(i), i)$ and $(S', \gamma') \cong (B\text{-}\mathit{Sph}^w(i), i)$. Thus, $(S, \gamma) \cong (S', \gamma')$. Moreover, $\mathit{col} = \mathit{col}' = \Phi(i)$.

(ii) Clearly, we have $\lambda(\alpha) = \lambda'(\alpha')$ and $\nu(\alpha) = \nu'(\alpha')$.

(iii) Suppose $S \cong S'$ ($S = S'$, for simplicity) and $\mathit{col} = \mathit{col}'$. According to the definition of q_i, there are positions i_1, i_2 of w such that $\mathit{dist}^w(i, i_1) \leq B$, $\mathit{dist}^w(i, i_2) \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_1), i)$, $(S, \alpha') \cong (B\text{-}\mathit{Sph}^w(i_2), i)$, and $\mathit{col} = \Phi(i_1) = \Phi(i_2)$. We have $(B\text{-}\mathit{Sph}^w(i_1), i) \cong (B\text{-}\mathit{Sph}^w(i_2), i)$. As i_1 and i_2 have a B-overlap, we also have, by Lemma 3, $i_1 = i_2$. We deduce $\alpha = \alpha'$.

Next, we define a tuple $t_i = (p_i, g_i) \xrightarrow{a_i} (q_i, f_i)$ for all $i \in [n]$. We let $(p_i)_\lhd = q_{\mathsf{prev}^w_\lhd(i)}$ (which might be undefined). Moreover, let g_i and f_i be uniquely given by conditions T7 and T8 where we replace q with q_i. Before we check that conditions (1)–(4) of a run are satisfied, we verify that t_i is indeed a transition. In the following, we let E always refer to $E = (S, \alpha, \mathit{col})$ with $S = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$.

T1 Obviously, we have $\mathit{label}(q_i) = a_i$.

T2 Let $\lhd \in \mathcal{S} \backslash \mathrm{dom}(p_i)$ (which implies that $\mathsf{prev}^w_\lhd(i)$ is undefined) and $E \in q_i$. We have $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$ for some i_c with $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$. As $\mathsf{prev}^w_\lhd(i)$ is undefined, we conclude that $\mathsf{prev}^E_\lhd(\alpha)$ is undefined, too.

T3 Let $\lhd \in \mathrm{dom}(p_i)$, $E \in q_i$, $j \in U$, and $i_\lhd = \mathsf{prev}^w_\lhd(i)$.

Suppose $j \lhd^E \alpha$. We need to show $E[j] \in q_{i_\lhd}$. As $E \in q_i$, there is $i_\mathrm{c} \in [n]$ such that $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$, and $\mathit{col} = \Phi(i_\mathrm{c})$. Since $\mathit{dist}^E(\gamma, j) \leq B$ implies $\mathit{dist}^w(i_\mathrm{c}, i_\lhd) \leq B$, and since $(S, j) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i_\lhd)$ and $\mathit{col} = \Phi(i_\mathrm{c})$, we deduce $E[j] = (S, j, \mathit{col}) \in q_{i_\lhd}$.

Conversely, suppose $E[j] \in q_{i_\lhd}$. We shall show $j \lhd^E \alpha$. There are positions $i_\mathrm{c}, i'_\mathrm{c} \in [n]$ such that we have $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$, $\mathit{dist}^w(i'_\mathrm{c}, i_\lhd) \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$, $(S, j) \cong (B\text{-}\mathit{Sph}^w(i'_\mathrm{c}), i_\lhd)$, and $\mathit{col} = \Phi(i_\mathrm{c}) = \Phi(i'_\mathrm{c})$. Note that i_c and i'_c have a B-overlap. By Lemma 3, $i_\mathrm{c} = i'_\mathrm{c}$. As, then, $(S, j) \cong (B\text{-}\mathit{Sph}^w(i'_\mathrm{c}), i_\lhd)$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i'_\mathrm{c}), i)$, and $i_\lhd \lhd^w i$, we can deduce $j \lhd^E \alpha$.

T4 is shown similarly to T3.

T5 Let $\lhd \in \mathrm{dom}(p_i)$ and $E \in q_i$ such that $\mathsf{prev}^E_\lhd(\alpha)$ is undefined. There is $i_\mathrm{c} \in [n]$ such that $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$ and $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$. Now, suppose $\mathit{dist}^E(\gamma, \alpha) < B$. But then, we also have $\mathit{dist}^w(i_\mathrm{c}, i) < B$ and $\mathsf{prev}^E_\lhd(\alpha)$ is defined, a contradiction. We deduce that $\mathit{dist}^E(\gamma, \alpha) = B$.

T6 is shown similarly to T5.

T7 and T8 are immediate.

So far, we know that t_i is a transition. Now, let us check the run conditions.

(1) and (2) are readily verified.

(3) Consider guard $g_i = g_1 \wedge g_2 \wedge g_3$. We first check subformula g_1. For $k_1, k_2 \in [m]$, by the definition of $\sim_{q_i}$ and $G(w)$, $k_1 \sim_{q_i} k_2$ iff $d_i^{k_1} = d_i^{k_2}$. Now, consider g_2 and an atomic subformula $k = (\lhd, (E, k))$ where $k \in [m]$, $E \in q$, and $\lhd \in \mathit{type}^-(\alpha)$. Set $i_\lhd = \mathsf{prev}^w_\lhd(i)$, which must indeed exist (by T2). As $E \in q_i$, there is $i_\mathrm{c} \in [n]$ such that $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$, and $\mathit{col} = \Phi(i_\mathrm{c})$. This implies $\mathit{dist}^w(i_\mathrm{c}, i_\lhd) \leq B$, and we obtain $\rho_{i_\lhd}((E, k)) = d_i^k$ so that g_2 also holds. Finally, we have to check g_3. Consider its subformula $(\lhd_1, (E[j], k)) = (\lhd_2, (E[j], k))$ where $k \in [m]$, $E \in q_i$, $j \in U$, and $\lhd_1, \lhd_2 \in \mathit{type}^-(\alpha)$. Let $i_1 = \mathsf{prev}^w_{\lhd_1}(i)$ and $i_2 = \mathsf{prev}^w_{\lhd_2}(i)$ (they both exist). Moreover, let $j_1 = \mathsf{prev}^E_{\lhd_1}(\alpha)$ and $j_2 = \mathsf{prev}^E_{\lhd_2}(\alpha)$. As $E \in q_i$, there is $i_\mathrm{c} \in [n]$ such that $\mathit{dist}^w(i_\mathrm{c}, i) \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i)$, and $\mathit{col} = \Phi(i_\mathrm{c})$. Due to the isomorphism, there is a unique $i' \in [n]$ such that $\mathit{dist}^w(i_\mathrm{c}, i') \leq B$ and $(S, j) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i')$. Moreover, we have $(S, j_1) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i_1)$ and $(S, j_2) \cong (B\text{-}\mathit{Sph}^w(i_\mathrm{c}), i_2)$. In particular, $\mathit{dist}^w(i_\mathrm{c}, i_1) \leq B$ and $\mathit{dist}^w(i_\mathrm{c}, i_2) \leq B$. We deduce $\rho_{i_1}((E[j], k)) = \rho_{i_2}((E[j], k)) = d_{i'}^k$. Thus, g_3 is satisfied.

(4) Let $(E, k) \in R$. We distinguish three cases.

- If there is $j \in U$ such that $E[j] \in q_i$, and $\mathit{type}^-(j) \neq \emptyset$, then we have $f_i((E,k)) = (\lhd, (E,k))$ with $\lhd = \lhd_{E[j]}$. Since $E[j] \in q_i$, there is a position $i_c \in [n]$ such that $\mathit{dist}^w(i_c, i) \leq B$, $(S, j) \cong (B\text{-}\mathit{Sph}^w(i_c), i)$, and $\mathit{col} = \Phi(i_c)$. Moreover, there is a unique position $i' \in [n]$ such that $\mathit{dist}^w(i_c, i') \leq B$ and $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_c), i')$. As $j_\lhd = \mathsf{prev}^E_\lhd(j)$ is defined, $i_\lhd = \mathsf{prev}^w_\lhd(i)$ is defined, too. Note that $(S, j_\lhd) \cong (B\text{-}\mathit{Sph}^w(i_c), i_\lhd)$ and $\mathit{dist}^w(i_c, i_\lhd) \leq B$. We obtain $\rho_i((E,k)) = d^k_{i'} = \rho_{i_\lhd}((E,k))$.

- If there is $j \in U$ such that $E[j] \in q_i$ and $\mathit{type}^-(j) = \emptyset$, then $f_i((E,k)) = (k, \mathit{dist}^E(\alpha, j))$. We show $\rho_i((E,k)) \in \mathfrak{D}^k_{B'}(i)$ where $B' = \mathit{dist}^E(\alpha, j)$. As $E[j] \in q_i$, there is $i_c \in [n]$ such that $\mathit{dist}^w(i_c, i) \leq B$, $(S, j) \cong (B\text{-}\mathit{Sph}^w(i_c), i)$, and $\mathit{col} = \Phi(i_c)$. Thus, there is a unique position $i' \in [n]$ such that $\mathit{dist}^w(i_c, i') \leq B$ and $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_c), i')$. We have $\mathit{dist}^w(i', i) \leq \mathit{dist}^E(\alpha, j)$, and we can deduce $\rho_i((E,k)) = d^k_{i'} \in \mathfrak{D}^k_{B'}(i)$.

- If there is no $j \in U$ such that $E[j] \in q_i$, then $f_i((E,k))$ is undefined. Therefore, $\rho_i((E,k))$ should be undefined, too. Suppose, towards a contradiction, that $\rho_i((E,k)) \in \mathfrak{D}$. Then, there are $i_c, i' \in [n]$ such that we have $\mathit{dist}^w(i_c, i) \leq B$, $\mathit{dist}^w(i_c, i') \leq B$, $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_c), i')$, and $\mathit{col} = \Phi(i_c)$, But then, there is a unique $j \in U$ such that $(S, j) \cong (B\text{-}\mathit{Sph}^w(i_c), i)$ so that $E[j] \in q_i$, which is a contradiction.

We conclude that ξ is a run. Let us quickly verify that it is accepting. Trivially, $\Phi = \mathit{true}$ is satisfied. Now suppose $\lhd \in \mathcal{S}$ and consider any position $i \in [n]$ such that $\mathsf{next}^w_\lhd(i)$ is undefined. We have to show that q_i is contained in $F_\lhd$, i.e., $\mathsf{next}^E_\lhd(\alpha)$ is undefined for all $E \in q_i$. So suppose $E \in q_i$. There is $i_c \in [n]$ such that $\mathit{dist}^w(i_c, i) \leq B$ and $(S, \alpha) \cong (B\text{-}\mathit{Sph}^w(i_c), i)$. As $\mathsf{next}^w_\lhd(i)$ is undefined, $\mathsf{next}^E_\lhd(\alpha)$ must be undefined, too.

Every run keeps track of spheres. In this part of the proof, we show that we can infer, from every accepting run of $\mathcal{A}_B$ on data word w, the spheres that occur in $G(w)$.

Let $w = (a_1, d_1) \dots (a_n, d_n) \in (\Sigma \times \mathfrak{D}^m)^*$ be a data word and $G(w) = ([n], (\lhd^w)_{\lhd \in \mathcal{S}}, \hat{\lambda}, \hat{\nu})$ its graph. Suppose $\xi = (q_1, \rho_1) \dots (q_n, \rho_n)$ is an accepting run of $\mathcal{A}_B$ on w with corresponding transitions $t_1, \dots, t_n$ where $t_i = (p_i, g_i) \xrightarrow{a_i} (q_i, f_i)$.

The following claim states that an arbitrarily long path of an extended sphere E that starts in its active node is faithfully simulated by w. It will turn out to be crucial that, hereby, the data values in registers of the form $(E[j], k)$ are invariant during that simulation.

Lemma 4. *Let $i \in [n]$ be some position, $e \geq 0$, and $E = (S, \alpha, \mathit{col}) \in q_i$ with $S = (U, (\lhd^E)_{\lhd \in \mathcal{S}}, \lambda, \nu, \gamma)$. Suppose there are $j_0, \dots, j_e \in U$ and $\lhd_1 \dots, \lhd_e \in \mathcal{S}$ such that $\alpha = j_0$ and, for all $z \in \{0, \dots, e-1\}$, $j_z \lhd^E_{z+1} j_{z+1}$ or $j_{z+1} \lhd^E_{z+1} j_z$. Then, there is a unique sequence $i = i_0, \dots, i_e \in [n]$ such that the following hold:*

- *for each* $z \in \{0,\dots,e-1\}$, $j_z \lhd^E_{z+1} j_{z+1}$ *implies* $i_z \lhd^w_{z+1} i_{z+1}$ *and* $j_{z+1} \lhd^E_{z+1} j_z$ *implies* $i_{z+1} \lhd^w_{z+1} i_z$
- *for each* $z \in \{0,\dots,e\}$, *we have* $E[j_z] \in q_{i_z}$, $\lambda(j_z) = a_{i_z}$, *and* $\nu(j_z) = \hat{\nu}(i_z)$
- *for each* $z \in \{1,\dots,e\}$, $k \in [m]$, *and* $j \in U$, *we have* $\rho_{i_0}((E[j],k)) = \rho_{i_z}((E[j],k))$
- *for each* $z \in \{0,\dots,e\}$ *and* $k \in [m]$, *we have that* $\rho_{i_z}((E[j_z],k)) = d^k_{i_z}$

Proof. We proceed by induction on e. Suppose $e = 0$. By T1 and guard g_1 of T7, $\lambda(\alpha) = a_i$ and $\nu(\alpha) = \hat{\nu}(i)$. Let $k \in [m]$ and suppose $\mathit{type}^-(\alpha) \neq \emptyset$. Then, $f_i((E,k)) = (\lhd,(E,k))$ where we let $\lhd = \lhd_E$. Thus, $\rho_i((E,k)) = \rho_{\mathsf{prev}^w_\lhd(i)}((E,k))$. By guard g_2 of T7, we have $\rho_i((E,k)) = d^k_i$. If $\mathit{type}^-(\alpha) = \emptyset$, then $\rho_i((E,k)) = d^k_i$ is due to the update $f_i((E,k)) = (k,0)$ (T8).

So let $e \geq 0$, $j_0,\dots,j_e,j_{e+1} \in U$, and $\lhd_1,\dots,\lhd_e,\lhd_{e+1} \in \mathcal{S}$ such that $\alpha = j_0$ and, for every $z \in \{0,\dots,e\}$, $j_z \lhd^E_{z+1} j_{z+1}$ or $j_{z+1} \lhd^E_{z+1} j_z$. Let $i_0,\dots,i_e \in [n]$ be the unique corresponding sequence with the required properties. We consider two cases:

- Assume $j_e \lhd^E_{e+1} j_{e+1}$. Then, $q_{i_e} \notin F_{\lhd_{e+1}}$ so that $\mathsf{next}^w_{\lhd_{e+1}}(i_e)$ is defined. We set $i_{e+1} = \mathsf{next}^w_{\lhd_{e+1}}(i_e)$.
  Due to T4, we have $E[j_{e+1}] \in q_{i_{e+1}}$. By T1 and guard g_1 of T7, we obtain $\lambda(j_{e+1}) = a_{i_{e+1}}$, and $\nu(j_{e+1}) = \hat{\nu}(i_{e+1})$.
  Let $k \in [m]$ and $j \in U$. Due to condition T8, $E[j_{e+1}] \in q_{i_{e+1}}$ implies that $f_{i_{e+1}}((E[j],k)) = (\lhd,(E[j],k))$ for some $\lhd \in \mathcal{S}$. Due to guard g_3 of condition T7, we have $\rho_{\mathsf{prev}^w_\lhd(i_{e+1})}((E[j],k)) = \rho_{i_e}((E[j],k))$. We can now deduce $\rho_{i_e}((E[j],k)) = \rho_{i_{e+1}}((E[j],k))$.
  Finally, let $k \in [m]$. We have $f_{i_{e+1}}((E[j_{e+1}],k)) = (\lhd,(E[j_{e+1}],k))$ where we let $\lhd = \lhd_{E[j_{e+1}]}$. Thus, $\rho_{i_{e+1}}((E[j_{e+1}],k)) = \rho_{\mathsf{prev}^w_\lhd(i_{e+1})}((E[j_{e+1}],k))$. By guard g_2 of T7, we obtain $\rho_{i_{e+1}}((E[j_{e+1}],k)) = d^k_{i_{e+1}}$.
- Assume $j_{e+1} \lhd^E_{e+1} j_e$. By T2, $\lhd_{e+1} \in \mathrm{dom}(p_i)$. Thus, there is (a unique) i_{e+1} such that $i_{e+1} \lhd^w_{e+1} i_e$.
  By T3, we have $E[j_{e+1}] \in q_{i_{e+1}}$. Moreover, $\lambda(j_{e+1}) = a_{i_{e+1}}$, and $\nu(j_{e+1}) = \hat{\nu}(i_{e+1})$.
  Let $k \in [m]$ and $j \in U$. By condition T8, we have $E[j_e] \in q_{i_e}$ implies $f_{i_e}((E[j],k)) = (\lhd,(E[j],k))$ for some $\lhd \in \mathcal{S}$. Due to guard g_3 of condition T7, we have $\rho_{\mathsf{prev}^w_\lhd(i_e)}((E[j],k)) = \rho_{i_{e+1}}((E[j],k))$. We deduce $\rho_{i_e}((E[j],k)) = \rho_{i_{e+1}}((E[j],k))$.
  Finally, let $k \in [m]$. We distinguish two cases. Suppose $\mathit{type}^-(j_{e+1}) \neq \emptyset$. Then, $f_{i_{e+1}}((E[j_{e+1}],k)) = (\lhd,(E[j_{e+1}],k))$ where we let $\lhd = \lhd_{E[j_{e+1}]}$. Thus, $\rho_{i_{e+1}}((E[j_{e+1}],k)) = \rho_{\mathsf{prev}^w_\lhd(i_{e+1})}((E[j_{e+1}],k))$. By guard g_2 of T7, we have $\rho_{i_{e+1}}((E[j_{e+1}],k)) = d^k_{i_{e+1}}$. If $\mathit{type}^-(j_{e+1}) = \emptyset$, then $\rho_{i_{e+1}}((E[j_{e+1}],k)) = d^k_{i_{e+1}}$ is due to the update $f_{i_{e+1}}((E[j_{e+1}],k)) = (k,0)$ (T8).

This concludes the proof of Lemma 4. □

By means of Lemma 4, we will show that spheres that are contained in states indeed occur in a data word. It will be used in combination with the following simple monotonicity fact, which follows easily from the definitions.

Next, we show that a sphere correctly simulates w and vice versa, which concludes the correctness proof for $\mathcal{A}_B$.

For $i \in [n]$, let $E_i = (S_i, \alpha_i, col_i)$ with $S_i := (U_i, (\lhd^{E_i})_{\lhd \in \mathbb{S}}, \lambda_i, \nu_i, \gamma_i)$ be the unique extended sphere from q_i such that $\gamma_i = \alpha_i$. In particular, $S_i = \pi(q_i)$.

Lemma 5. *For all $i \in [n]$, we have B-$Sph^w(i) \cong S_i$.*

Proof. For $e \in \{0, \ldots, B\}$, let e-S_i denote the e-sphere of $(U_i, (\lhd^{E_i})_{\lhd \in \mathbb{S}}, \lambda_i, \nu_i)$ around γ_i, which is defined in the canonical manner. We show, by induction, the following more general statement:

For every $e \in \{0, \ldots, B\}$, there is an isomorphism $h : e\text{-}Sph^w(i) \to e\text{-}S_i$ such that, for each $i' \in [n]$ with $dist^w(i, i') \leq e$, we have $E_i[h(i')] \in q_{i'}$. (*)

We easily verify that (*) holds for $e = 0$. Now suppose there is an isomorphism $h : e\text{-}Sph^w(i) \to e\text{-}S_i$ with $e < B$. We extend the domain of h to elements i' with $dist^w(i, i') = e + 1$ as follows. Let $i_1, i_2 \in [n]$ such that $dist^w(i, i_1) = e$ and $dist^w(i, i_2) = e + 1$. Let $\lhd \in \mathbb{S}$. We distinguish several cases:

- Suppose $i_1 \lhd^w i_2$. Since $dist^w(i, i_1) < B$, we have $dist^w(\gamma_i, h(i_1)) < B$. By T6, there is $j_2 \in U_i$ such that $h(i_1) \lhd^{E_i} j_2$. Since $E_i[h(i_1)] \in q_{i_1}$, we obtain, by T1, T4, and T7, $\lambda_i(j_2) = a_{i_2}$, $\nu_i(j_2) = \hat{\nu}(i_2)$, and $E_i[j_2] \in q_{i_2}$.
- Suppose $i_2 \lhd^w i_1$. Similarly, due to $dist^w(i, i_1) < B$ and T5, there is $j_2 \in U_i$ such that $j_2 \lhd^{E_i} h(i_1)$. Using T1, T3, and T7, we obtain $\lambda_i(j_2) = a_{i_2}$, $\nu_i(j_2) = \hat{\nu}(i_2)$, and $E_i[j_2] \in q_{i_2}$.

We set $\bar{h}(i_2) = j_2$ and $\bar{h}(i') = h(i')$ for all positions i' in e-$Sph^w(i)$. In doing so, we extend the domain of h to elements with distance $e + 1$ from i. Note that this extension $\bar{h} : (e+1)\text{-}Sph^w(i) \to (e+1)\text{-}S_i$ is well defined, i.e., j_2 is uniquely determined by i_2 and does not depend on the choice of i_1 or $\lhd$: if, for i_2, we obtained distinct elements j_2 and j_2', then $E_i[j_2] \in q_{i_2}$ and $E_i[j_2'] \in q_{i_2}$, which contradicts the definition of a state.

We show that we obtain a homomorphism $\bar{h} : (e+1)\text{-}Sph^w(i) \to (e+1)\text{-}S_i$. Let $i_1, i_2 \in [n]$ such that $dist^w(i, i_1) = dist^w(i, i_2) = e + 1$. Moreover, let $\lhd \in \mathbb{S}$. Suppose $i_1 \lhd^w i_2$ (the case $i_2 \lhd^w i_1$ is symmetric). We have $E_i[\bar{h}(i_1)] \in q_{i_1}$ and $E_i[\bar{h}(i_2)] \in q_{i_2}$. By T3 (or T4), this implies $\bar{h}(i_1) \lhd^{E_i} \bar{h}(i_2)$.

Next, we show that $\bar{h}$ is surjective. Let $j_1, j_2 \in U_i$ and $\lhd \in \mathbb{S}$ such that $dist^{E_i}(\gamma_i, j_1) = e$, $dist^{E_i}(\gamma_i, j_1) = e + 1$, and $j_1 \lhd^{E_i} j_2$ (the case $j_2 \lhd^{E_i} j_1$ is similar). We have $E_i[j_1] \in q_{h^{-1}(j_1)}$. By T4 and $q_{h^{-1}(j_1)} \notin F_\lhd$, there is $i_2 \in [n]$ such that $dist^w(i, i_2) = e + 1$, $h^{-1}(j_1) \lhd^w i_2$, and $E_i[j_2] \in q_{i_2}$. We deduce that $\bar{h}$ is surjective.

Let us show that $\bar{h}$ is injective. Let $i_1, i_2 \in [n]$ such that $dist^w(i, i_1) = dist^w(i, i_2) = e + 1$. Assume $i_1 \neq i_2$. We show that, then, $\bar{h}(i_1) \neq \bar{h}(i_2)$. Let $j_1 = \bar{h}(i_1)$ and $j_2 = \bar{h}(i_2)$. Assume, towards a contradiction, that $j_1 = j_2$. Furthermore, assume $i_1 < i_2$ (the other case is symmetric). In E_i, there are paths from j_1 to α and from α to j_1 that are simulated, in w, by paths from i_2 to i and from i to i_1, respectively. By Lemma 4 and monotonicity of a signature, we can simulate these paths of E_i arbitrarily often in w. This yields an infinite

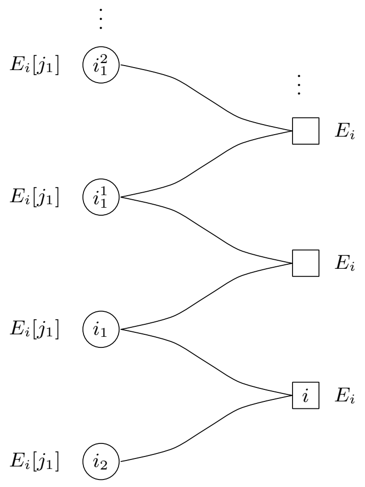

Fig. 7. $\bar{h}$ is injective

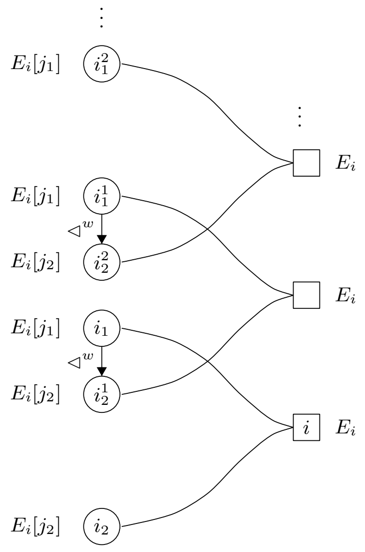

Fig. 8. $\bar{h}^{-1}$ is a homomorphism

descending chain $\ldots < i_1^2 < i_1^1 < i_1 < i_2$ such that $E[j_1] \in q_{i_1^l}$ and $d_{i_2}^k = d_{i_1}^k = d_{i_1^l}^k$ for all $l \geq 1$ and $k \in [m]$. But this is a contradiction, as every word position has only finitely many smaller positions. The procedure is illustrated in Figure 7.

Finally, we show that $\bar{h} : (e+1)\text{-}Sph^w(i) \to (e+1)\text{-}S_i$ is actually an isomorphism. Let $j_1, j_2 \in U_i$ and $\lhd \in \mathcal{S}$ such that $dist^{E_i}(\gamma, j_1) = dist^{E_i}(\gamma, j_2) = e+1$ and $j_1 \lhd^{E_i} j_2$. We show that this implies $\bar{h}^{-1}(j_1) \lhd^w \bar{h}^{-1}(j_2)$. Set $i_1 = \bar{h}^{-1}(j_1)$ and $i_2 = \bar{h}^{-1}(i_2)$. Assume, towards a contradiction, that $i_1 \not\lhd^w i_2$. We have $j_1 \neq j_2$, $E_i[j_1] \in q_{i_1}$, and $E_i[j_2] \in q_{i_2}$. Due to the definition of the set of states of $\mathcal{A}_B$, this implies $i_1 \neq i_2$. Suppose $\mathsf{next}_{\lhd}^{w}(i_1) < i_2$ (the other case is similar). Again, by Lemma 4 and monotonicity of $\lhd^w$, we can build an infinite descending chain $\ldots < i_1^2 < i_1^1 < i_1 < i_2$ such that $E[j_1] \in q_{i_1^l}$ for all $l \geq 1$ (cf. Figure 8). This is a contradiction. □

B. Comparison with Class Automata

We compare class register automata to class automata [3], which have been shown to capture all (extended) XPath queries. Class automata are a smooth (undecidable) extension of data automata and, therefore, of class memory automata. A class automaton is suitable to work over words (even trees) with multiple data values. It consists in a pair $(\mathcal{A}, \mathcal{B})$ where $\mathcal{A}$ is a non-deterministic letter-to-letter transducer from the label alphabet Σ to some working alphabet Γ, and $\mathcal{B}$ is a finite automaton over $\Gamma \times \{0,1\}^m$. A data word $(a_1, d_1) \ldots (a_n, d_n) \in \Sigma \times \{0,1\}^m$ is accepted if, for input $a_1 \ldots a_n$, there is some output $u_1 \ldots u_n \in \Gamma^*$ of $\mathcal{A}$ such that, for all $d \in \mathfrak{D}$, the word $(u_1, b_1) \ldots (u_n, b_n) \in (\Gamma \times \{0,1\}^m)^*$ is accepted by $\mathcal{B}$. Hereby, $b_i^k = 1$ iff $d_i^k = d$.

We will show that, for $m = 2$, class automata capture neither EMSO logic nor non-guessing class register automata. Note that class automata do not depend on a signature. To allow for a fair comparison, we choose the simple signature $\mathcal{S}^2_{+1,\sim} = \{\prec_{+1}, \prec^1_{\sim}, \prec^2_{\sim}\}$.

Theorem 7. *There is $L \in \mathrm{r}\mathbb{EMSO}(\mathcal{S}^2_{+1,\sim}) \cap \mathbb{CRA}^-(\mathcal{S}^2_{+1,\sim})$ such that L cannot be recognized by any class automaton.*

Proof. Let $\Sigma = \{a\}$ and $\mathfrak{D} = \mathbb{N}$. Using [3], one can show that there is no class automaton that recognizes $L = [\{(a,1,1)\ldots(a,n,n)(a,1,1)\ldots(a,n,n) \mid n \geq 1\}]_{\mathcal{S}^2_{+1,\sim}}$. It is, however, easy to define an rEMSO($\mathcal{S}^2_{+1,\sim}$)-sentence for L. We restrict to the construction of a non-guessing class register automaton, which is very similar to the automaton from Example 5. Here, we will need four registers, r^k_1 and r^k_2 for $k = 1, 2$. The crucial difference is in the second phase, where we encounter a data value for the second time. We henceforth require that, at position $n+i$, the k-th data value d^k_{n+i} is contained in register r^k_1 at $\mathsf{prev}_{\prec^k_\sim}(n+i) = i$. The value d^k_{n+i} is henceforth stored in r^k_1 and has to coincide, at position $n+i+1$, with the contents of r^k_2 at position $\mathsf{prev}_{\prec^k_\sim}(n+i+1) = i+1$. $\square$