Local First-Order Logic with Two Data Values

Benedikt Bollig
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF

Arnaud Sangnier
IRIF, Université de Paris, CNRS

Olivier Stietel
IRIF, Université de Paris, CNRS
Université Paris-Saclay, CNRS, ENS Paris-Saclay, LMF

Abstract

We study first-order logic over unordered structures whose elements carry two data values from an infinite domain. Data values can be compared wrt. equality so that the formalism is suitable to specify the input-output behavior of various distributed algorithms. As the logic is undecidable in general, we introduce a family of local fragments that restrict quantification to neighborhoods of a given reference point. Our main result establishes decidability of the satisfiability problem for one of these non-trivial local fragments. On the other hand, already slightly more general local logics turn out to be undecidable. Altogether, we draw a landscape of formalisms that are suitable for the specification of systems with data and open up new avenues for future research.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases first-order logic, data values, specification of distributed algorithms

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2021.29

Related Version *A full version of the paper containing missing proofs is available via the following link*: `https://hal.archives-ouvertes.fr/hal-03353214`

Funding Partly supported by ANR FREDDA (ANR-17-CE40-0013).

Acknowledgements The authors would like to thank the reviewers for their careful reading of the paper and their helpful comments.

1 Introduction

Data logics have been introduced to reason about structures whose elements are labeled with a value from an infinite alphabet (e.g., XML documents) [26]. Expressive decidable fragments include notably two-variable logics over data words and data trees [5, 6]. The decidability frontier is fragile, though. Extensions to two data values, for example, quickly lead to an undecidable satisfiability problem. From a modeling point of view, those extensions still play an important role. When specifying the *input-output behavior* of distributed algorithms [13, 23], processes get an input value and produce an output value, which requires two data values per process. In leader election or renaming algorithms, for instance, a process gets its unique identifier as input, and it should eventually output the identifier of a common leader (leader election) or a unique identifier from a restricted name space (renaming).

In this paper, we consider a natural extension of first-order logic over unordered structures whose elements carry two data values from an infinite domain. There are two major differences between most existing formalisms and our language. While previous data logics are usually interpreted over words or trees, we consider unordered structures (or multisets). When each element of such a structure represents a process, we therefore do not assume a particular processes architecture, but rather consider clouds of computing units. Moreover, decidable data logics are usually limited to one value per element, which would not be sufficient to model an input-output relation. Hence, our models are algebraic structures consisting of a universe and functions assigning to each element two integers. We remark that, for many distributed algorithms, the precise data values are not relevant, but whether or not they are the same is. Like [5, 6], we thus add binary relations that allow us to test if two data values are identical and, for example, whether all output values were already present in the collection of input values (as required for leader election).

The first fundamental question that arises is whether a given specification is consistent. This leads us to the satisfiability problem. While the general logic considered here turns out to be undecidable already in several restricted settings, our main result shows that an interesting fragment preserves decidability. The fragment is a *local* logic in the sense that data values can only be compared within the direct neighborhood of a (quantified) reference process. The first value at the reference point can be compared with any second value in the neighborhood in terms of what we call the *diagonal relation*. In this work, we do not allow the symmetrical relation, but we hope we could adapt our technique to this case as well.

However, we do not restrict comparisons of first values with each other in a neighborhood, nor do we restrict comparisons of second values with each other. Note that adding only one diagonal relation still constitutes an extension of the (decidable) two-variable first-order logic with two equivalence relations [18–20]: equivalence classes consist of those elements with the same first value, respectively, second value. In fact, our main technical contribution is a reduction to this two-variable logic. The reduction requires a careful relabelling of the underlying structures so as to be able to express the diagonal relation in terms of the two equivalence relations. In addition, the reduction takes care of the fact that our logic does not restrict the number of variables. We can actually count elements up to some threshold and express, for instance, that at most five processes crash (in the context of distributed algorithms). This is a priori not possible in two-variable logic.

More Related Work. Orthogonal extensions for multiple data values include shuffle expressions for nested data [3] and temporal logics [10, 17]. Other generalizations of data logics allow for an order on data values [24, 27]. The application of formal methods in the context of distributed algorithms is a rather recent but promising approach (cf. for a survey [21]). A particular branch is the area of parameterized systems, which, rather than on data, focuses on the (unbounded) number of processes as the parameter [4, 12]. Other related work includes [11], which considers temporal logics involving quantification over processes but without data, while [1] introduces an (undecidable) variant of propositional dynamic logic that allows one to reason about totally ordered process identifiers in ring architectures. First-order logics for *synthesizing* distributed algorithms were considered in [7, 14]. A counting extension of two-variable first-order logic over finite data words with one data value per position has been studied in [2].

Outline. Section 2 introduces basic notions such as structures and first-order logic, and our local first-order logic and the associated satisfiability problem(s). We identify and solve the decidable case in Section 3. In Section 4, we show that minor extensions of our logic result in undecidability. We conclude in Section 5. Some proof details can be found in the full version of this paper, which is available at: `https://hal.archives-ouvertes.fr/hal-03353214`

2 Structures and First-Order Logic

2.1 Structures and First-Order Logic

Let Σ be a finite set of unary relation symbols, sometimes called unary predicates. A *data structure* over Σ is a tuple $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)_{\sigma\in\Sigma})$ (in the following, we simply write $(A, f_1, f_2, (P_\sigma)))$ where A is a nonempty finite set, $P_\sigma \subseteq A$ for all $\sigma \in \Sigma$, and f_1 and f_2 are mappings $A \to \mathbb{N}$ assigning a *data value* to each element. We let $\mathit{Val}_\mathfrak{A} = \{f_1(a) \mid a \in A\} \cup \{f_2(a) \mid a \in A\}$. The set of all data structures over Σ is denoted by $\mathrm{Data}[\Sigma]$.

While this representation of data structures is often very convenient to refer to the first or second data value of an element, a more standard way of representing mathematical structures is in terms of binary relations. For every $(i, j) \in \{1, 2\} \times \{1, 2\}$, the mappings f_1 and f_2 determine a binary relation ${}_i{\sim}_j^\mathfrak{A} \subseteq A \times A$ as follows: $a \;{}_i{\sim}_j^\mathfrak{A}\; b$ if $f_i(a) = f_j(b)$. We may omit the superscript $\mathfrak{A}$ if it is clear from the context. This representation is particularly useful when we consider logics as specification languages.

Let $\Gamma \subseteq \{1, 2\} \times \{1, 2\}$ be a set of binary relation symbols, which determines the binary relation symbols ${}_i{\sim}_j$ at our disposal, and let $\mathcal{V} = \{x, y, \ldots\}$ be a countably infinite set of variables. The set $\mathrm{FO}[\Sigma; \Gamma]$ of first-order formulas interpreted over data structures over Σ is inductively given by the grammar $\varphi ::= \sigma(x) \mid x \;{}_i{\sim}_j\; y \mid x = y \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x.\varphi$, where x and y range over $\mathcal{V}$, σ ranges over Σ, and $(i, j) \in \Gamma$. We use standard abbreviations such as $\wedge$ for conjunction and $\to$ for implication. We write $\varphi(x_1, \ldots, x_n)$ to indicate that the free variables of φ are among $x_1, \ldots, x_n$. A formula without free variables is called a *sentence*.

For $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Sigma]$ and a formula $\varphi \in \mathrm{FO}[\Sigma; \Gamma]$, the satisfaction relation $\mathfrak{A} \models_I \varphi$ is defined wrt. an interpretation function $I : \mathcal{V} \to A$. The purpose of I is to assign an interpretation to every (free) variable of φ so that φ can be given a truth value. For $x \in \mathcal{V}$ and $a \in A$, the interpretation function $I[x/a]$ maps x to a and coincides with I on all other variables. We then define:

$$\begin{array}{ll}
\mathfrak{A} \models_I \sigma(x) \text{ if } I(x) \in P_\sigma & \mathfrak{A} \models_I \varphi_1 \vee \varphi_2 \text{ if } \mathfrak{A} \models_I \varphi_1 \text{ or } \mathfrak{A} \models_I \varphi_2 \\
\mathfrak{A} \models_I x \;{}_i{\sim}_j\; y \text{ if } I(x) \;{}_i{\sim}_j^\mathfrak{A}\; I(y) & \mathfrak{A} \models_I \neg\varphi \text{ if } \mathfrak{A} \not\models_I \varphi \\
\mathfrak{A} \models_I x = y \text{ if } I(x) = I(y) & \mathfrak{A} \models_I \exists x.\varphi \text{ if there is } a \in A \text{ with } \mathfrak{A} \models_{I[x/a]} \varphi
\end{array}$$

Finally, for a sentence φ (without free variables), we write $\mathfrak{A} \models \varphi$ if there exists an interpretation function I such that $\mathfrak{A} \models_I \varphi$.

▶ **Example 1.** Assume a unary predicate leader $\in \Sigma$ and $(1, 2) \in \Gamma$. We use the first data value to denote the input of a distributed algorithm and the second data value to denote the output. The following formula from $\mathrm{FO}[\Sigma; \Gamma]$ expresses correctness of a leader-election algorithm: (i) there is a unique process that has been elected leader, and (ii) all processes agree, in terms of their output values, on the identity (the input value) of the leader: $\exists^{=1}x.\mathrm{leader}(x) \wedge \forall y.\exists x.(\mathrm{leader}(x) \wedge x \;{}_1{\sim}_2\; y)$. Here $\exists^{=1}x$ is a shortcut for "there exists exactly one x". Its definition is provided later on. ◇

Note that every choice of Γ gives rise to a particular logic, whose formulas are interpreted over data structures over Σ. Instead of $\mathrm{FO}[\Sigma; \{(1, 1), (2, 2)\}]$, we may also simply write $\mathrm{FO}[\Sigma; (1, 1), (2, 2)]$ and so on. We will focus on the satisfiability problem for these logics. Let $\mathcal{F}$ denote a generic class of first-order formulas, parameterized by Σ and Γ. In particular, for $\mathcal{F} = \mathrm{FO}$, we have that $\mathcal{F}[\Sigma; \Gamma]$ is the class $\mathrm{FO}[\Sigma; \Gamma]$.

▶ **Definition 2.** *The problem* $\mathrm{DataSat}(\mathcal{F}, \Gamma)$ *for* $\mathcal{F}$ *and* Γ *is defined as follows: Given a finite set* Σ *and a sentence* $\varphi \in \mathcal{F}[\Sigma; \Gamma]$*, is there* $\mathfrak{A} \in \mathrm{Data}[\Sigma]$ *such that* $\mathfrak{A} \models \varphi$ *?*

The following negative result, which was shown in [16, Theorem 1], calls for restrictions of the general logic:

▶ **Theorem 3** ([16]). DataSat(FO, $\{(1,1),(2,2)\}$) *is undecidable, even when requiring that* $\Sigma = \emptyset$.

A Normal Form. When $\Gamma = \emptyset$, satisfiability of monadic first-order logic is decidable [9, Corollary 6.2.2] and the logic actually has a useful normal form. Let $\varphi(x_1,\ldots,x_n,y) \in \mathrm{FO}[\Sigma;\emptyset]$ and $k \geq 1$ be a natural number. We use $\exists^{\geq k} y.\varphi(x_1,\ldots,x_n,y)$ as an abbreviation for $\exists y_1 \ldots \exists y_k. \bigwedge_{1\leq i<j\leq k} \neg(y_i = y_j) \wedge \bigwedge_{1\leq i\leq k} \varphi(x_1,\ldots,x_n,y_i)$. Thus, $\exists^{\geq k} y.\varphi$ says that there are at least k distinct elements y that verify φ. We call a formula of the form $\exists^{\geq k} y.\varphi$ a *threshold formula*. We also use $\exists^{=k} y.\varphi$ as an abbreviation for $\exists^{\geq k} y.\varphi \wedge \neg\exists^{\geq k+1} y.\varphi$.

When $\Gamma = \emptyset$, the out-degree of every element is 0 so that, over this particular signature, we deal with structures of bounded degree. The following lemma will turn out to be useful. It is due to Hanf's locality theorem [15,22] for structures of bounded degree (cf. [8, Theorem 2.4]).

▶ **Lemma 4.** *Every formula from* $\mathrm{FO}[\Sigma;\emptyset]$ *with one free variable* x *is effectively equivalent to a Boolean combination of formulas of the form* $\sigma(x)$ *with* $\sigma \in \Sigma$ *and threshold formulas of the form* $\exists^{\geq k} y.\varphi_U(y)$ *where* $U \subseteq \Sigma$ *and* $\varphi_U(y) = \bigwedge_{\sigma\in U} \sigma(y) \wedge \bigwedge_{\sigma\in\Sigma\setminus U} \neg\sigma(y)$.

Extended Two-Variable First-Order Logic. An orthogonal way to obtain decidability is to restrict to two variables and $\Gamma = \{(1,1),(2,2)\}$. The two-variable fragment $\mathrm{FO}^2[\Sigma;\Gamma]$ contains all $\mathrm{FO}[\Sigma;\Gamma]$ formulas that use only two variables (usually x and y). In a two-variable formula, however, each of the two variables can be used arbitrarily often. The satisfiability problem of two-variable logic over arbitrary finite structures with two equivalence relations is decidable [20, Theorem 15]. By a straightforward reduction to this problem, we obtain:

▶ **Theorem 5** ([20]). *The problem* DataSat(FO^2, $\{(1,1),(2,2)\}$) *is decidable.*

Actually, this result can be generalized to *extended* two-variable first-order logic. A formula belongs to ext-$\mathrm{FO}^2[\Sigma,\Gamma]$ if it is of the form $\varphi \wedge \psi$ where $\varphi \in \mathrm{FO}[\Sigma;\emptyset]$ and $\psi \in \mathrm{FO}^2[\Sigma,\Gamma]$. To obtain the next result, the idea consists in first translating the formula $\varphi \in \mathrm{FO}[\Sigma;\emptyset]$ to a two-variable formula thanks to new unary predicates.

▶ **Proposition 6.** *The problem* DataSat(ext-FO^2, $\{(1,1),(2,2)\}$) *is decidable.*

2.2 Local First-Order Logic

We are interested in logics that combine the advantages of the logics considered so far, while preserving decidability. With this in mind, we will study *local* logics, where the scope of quantification is restricted to the neighborhood of a given element.

The neighborhood of an element a includes all elements whose distance to a is bounded by a given radius. It is formalized using the notion of a Gaifman graph (for an introduction, see [22]). In fact, we use a variant that is suitable for our setting and that we call *data graph*. Fix sets Σ and Γ. Given a data structure $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Sigma]$, we define its *data graph* $\mathcal{G}(\mathfrak{A}) = (V_{\mathcal{G}(\mathfrak{A})}, E_{\mathcal{G}(\mathfrak{A})})$ with set of vertices $V_{\mathcal{G}(\mathfrak{A})} = A \times \{1,2\}$ and set of edges $E_{\mathcal{G}(\mathfrak{A})} = \{((a,i),(b,j)) \in V_{\mathcal{G}(\mathfrak{A})} \times V_{\mathcal{G}(\mathfrak{A})} \mid a = b \text{ and } i \neq j, \text{ or } (i,j) \in \Gamma \text{ and } a \;{}_i{\sim}_j\; b\}$. The graph $\mathcal{G}(\mathfrak{A})$ is illustrated in Figure 1.

We define the distance $d^{\mathfrak{A}}((a,i),(b,j)) \in \mathbb{N} \cup \{\infty\}$ between two elements (a,i) and (b,j) from $A \times \{1,2\}$ as the length of the shortest *directed* path from (a,i) to (b,j) in $\mathcal{G}(\mathfrak{A})$. In fact, as the graph is directed, the distance function might not be symmetric. For $a \in A$ and

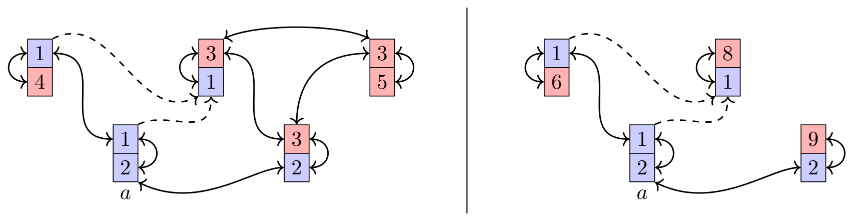

Figure 1 On the left: A data structure $\mathfrak{A}$ and its data graph $\mathcal{G}(\mathfrak{A})$ when $\Gamma = \{(1,1),(2,2),(1,2)\}$. Unidirectional edges are dashed. The blue nodes represent $B_1^{\mathfrak{A}}(a)$. On the right is $\mathfrak{A}|_a^1$.

$r \in \mathbb{N}$, the *radius-r-ball around a* is the set $B_r^{\mathfrak{A}}(a) = \{(b,j) \in V_{\mathcal{G}(\mathfrak{A})} \mid d^{\mathfrak{A}}((a,i),(b,j)) \leq r$ for some $i \in \{1,2\}\}$. That is, it contains the elements of $V_{\mathcal{G}(\mathfrak{A})}$ that can be reached from $(a,1)$ or $(a,2)$ through a directed path of length at most r. In the left-hand side of Figure 1, $B_1^{\mathfrak{A}}(a)$ is given by the blue nodes.

Consider an injective mapping $\pi : A \times \{1,2\} \to \mathbb{N} \setminus \mathit{Val}_{\mathfrak{A}}$. We define the *$r$-neighborhood of a in $\mathfrak{A}$* as the structure $\mathfrak{A}|_a^r = (A', f_1', f_2', (P_\sigma')) \in \mathrm{Data}[\Sigma]$. Its universe is $A' = \{b \in A \mid (b,i) \in B_r^{\mathfrak{A}}(a)$ for some $i \in \{1,2\}\}$. Moreover, $f_i'(b) = f_i(b)$ if $(b,i) \in B_r^{\mathfrak{A}}(a)$, and $f_i'(b) = \pi((b,i))$ otherwise. Finally, P_σ' is the restriction of P_σ to A'. To illustrate this definition, we use again Figure 1. The structure $\mathfrak{A}|_a^1$ is given by the four elements that contain at least one blue node. However, the values of the red nodes have to be replaced by pairwise distinct fresh values not contained in $\{1,\ldots,5\}$. Note that the precise values do not matter.

We are now ready to present the logic r-Loc-FO$[\Sigma;\Gamma]$, where $r \in \mathbb{N}$, interpreted over structures from Data$[\Sigma]$. It is given by the grammar

$$\varphi \ ::= \ \langle\!\langle \psi \rangle\!\rangle_x^r \ \mid \ x = y \ \mid \ \exists x.\varphi \ \mid \ \varphi \vee \varphi \ \mid \ \neg\varphi$$

where ψ is a formula from FO$[\Sigma;\Gamma]$ with (at most) one free variable x. For $\mathfrak{A} \in \mathrm{Data}[\Sigma]$ and interpretation function I, we define $\mathfrak{A} \models_I \langle\!\langle \psi \rangle\!\rangle_x^r$ if $\mathfrak{A}|_{I(x)}^r \models_I \psi$.

▶ **Example 7.** We can rewrite the formula from Example 1 so that it falls into the fragment 1-Loc-FO$[\Sigma;(1,1),(2,2),(2,1)]$: $\exists^{=1}x.\langle\!\langle \mathrm{leader}(x) \rangle\!\rangle_x^1 \wedge \forall y.\langle\!\langle \exists x.\mathrm{leader}(x) \wedge y \ {}_2{\sim}_1\ x \rangle\!\rangle_y^1$. The next formula specifies an algorithm in which all processes suggest a value and then choose a new value among those that have been suggested at least three times: $\forall x.\langle\!\langle \exists^{\geq 3} y. x \ {}_2{\sim}_1\ y \rangle\!\rangle_x^1$. We can also specify partial renaming, i.e., two output values agree only if their input values are the same: $\forall x.\langle\!\langle \forall y.(x \ {}_2{\sim}_2\ y \to x \ {}_1{\sim}_1\ y) \rangle\!\rangle_x^1$. Conversely, $\forall x.\langle\!\langle \forall y.(x \ {}_1{\sim}_1\ y \to x \ {}_2{\sim}_2\ y) \rangle\!\rangle_x^1$ specifies partial fusion of equivalences classes. ◇

3 Decidability With One Diagonal Relation

We will show in this section that DataSat(1-Loc-FO, $\{(1,1),(2,2),(1,2)\}$) (or, symmetrically, DataSat(1-Loc-FO, $\{(1,1),(2,2),(2,1)\}$)) is decidable. To this end, we will give a reduction to DataSat(ext-FO^2, $\{(1,1),(2,2)\}$). The rest of this section is devoted to this reduction.

Henceforth, we fix a finite set Σ as well as $\Gamma = \{(1,1),(2,2),(1,2)\}$ and the *diagonal-free set* $\Gamma_{df} = \{(1,1),(2,2)\}$. Moreover, we let Θ range over arbitrary finite sets such that $\Sigma \subseteq \Theta$ and $\Theta \cap \{\mathsf{eq},\mathsf{ed}\} = \emptyset$, where eq and ed are special unary symbols that are introduced below.

We start with some crucial notion. Suppose $\Gamma' \subseteq \Gamma$ (which will later be instantiated by either Γ_{df} or Γ). Consider a data structure $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Theta]$ with $\Sigma \subseteq \Theta$. Given $U \subseteq \Sigma$ and a nonempty set $R \subseteq \Gamma'$, the *environment* of $a \in A$ is defined as

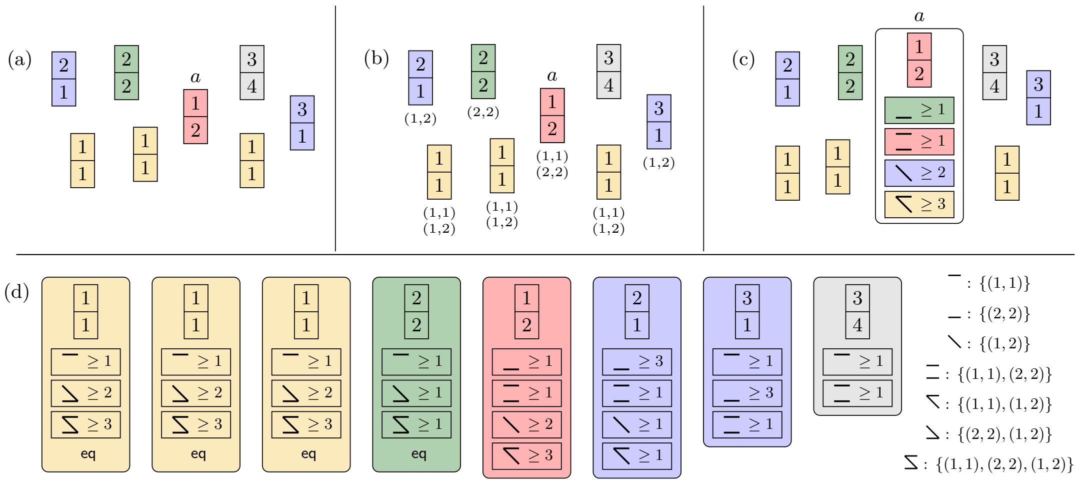

Figure 2 (a) A data structure over $\Sigma = \emptyset$. (b) Adding unary predicates for a given element a. (c) Adding counting constraints to a. (d) A well-typed data structure from $\text{Data}[\{\texttt{eq}\} \cup \mathbb{C}_3]$.

$$\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma'}(a,U,R) = \big\{ b \in A \mid U = \{\sigma \in \Sigma \mid b \in P_\sigma\} \text{ and } R = \{(i,j) \in \Gamma' \mid a \;{}_i{\sim}_j^{\mathfrak{A}}\; b\} \big\} .$$

Thus, it contains the elements that carry exactly the labels from U (relative to Σ) and to which a is related precisely in terms of the relations in R (relative to Γ').

▶ **Example 8.** Consider $\mathfrak{A} \in \text{Data}[\Sigma]$ from Figure 2(a) where $\Sigma = \emptyset$. Then, the set $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, \{(1,1),(1,2)\}) = \texttt{Env}_{\mathfrak{A},\Sigma,\Gamma_{df}}(a, \emptyset, \{(1,1)\})$ contains exactly the yellow elements (with data-value pairs $(1,1)$), and $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, \{(1,2)\})$ contains the two blue elements (with data-value pairs $(2,1)$ and $(3,1)$). ◇

Let us now go through the reduction step by step.

Step 1: Transform Binary into Unary Relations

In the first step, we get rid of the binary relations by representing them as unary ones. In fact, in a formula $\langle\!\langle \psi \rangle\!\rangle_x^1$ from 1-Loc-FO$[\Sigma;\Gamma]$, ψ only talks about elements that are directly related to $a = I(x)$ in terms of pairs from Γ. In fact, we can rewrite ψ into ψ' so that all comparisons are wrt. x, i.e., they are of the form $x \;{}_i{\sim}_j\; y$. Then, a pair $(i,j) \in \Gamma$ can be seen as a unary predicate that holds at b iff $a \;{}_i{\sim}_j\; b$. In this way, we eliminate the binary relations and replace ψ' with a first-order formula ψ'' over unary predicates.

▶ **Example 9.** Adding unary relations to a data structure for a given element a is illustrated in Figure 2(b) (recall that $\Sigma = \emptyset$). ◇

Thanks to the unary predicates, we can now apply Lemma 4 (which was a consequence of locality of first-order logic over unary symbols only). That is, to know whether ψ'' holds when x is interpreted as a, it is enough to know how often every unary predicate is present in the environment of a, counted only up to some $M \geq 1$. However, we will then give up the information of whether the two data values at a coincide or not. Therefore, we introduce a unary predicate $\texttt{eq}$, which shall label those events whose two data values coincide. Accordingly, we say that $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \text{Data}[\Theta \cup \{\texttt{eq}\}]$ is *eq-respecting* if, for all $a \in A$, we have $a \in P_{\texttt{eq}}$ iff $f_1(a) = f_2(a)$.

Once we add this information to a, it is enough to know the size of $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a,U,R)$ for every $U \subseteq \Sigma$ and nonempty $R \subseteq \Gamma$, measured up to M. To reason about these

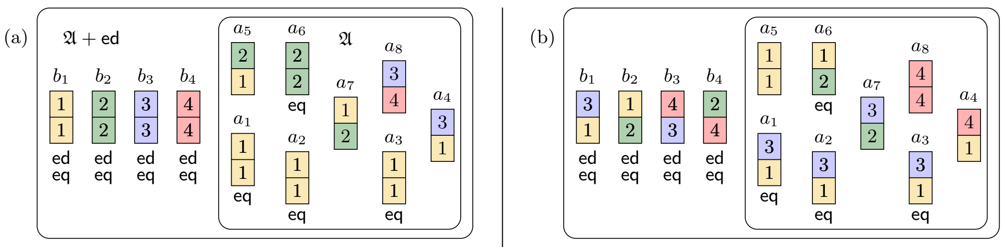

Figure 3 (a) Adding diagonal elements. (a)←(b) Making a data structure eq-respecting.

sizes, we introduce a unary predicate $\wr U, R, m \wr$ for all $U \subseteq \Sigma$, nonempty sets $R \subseteq \Gamma$, and $m \in \{1, \ldots, M\}$ (which is interpreted as "$\geq m$"). We also call such a predicate a *counting constraint* and denote the set of all counting constraints by $\mathtt{C}_M$ (recall that we fixed Σ and Γ). For a finite set Θ with $\Sigma \subseteq \Theta$, we call $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Theta \cup \mathtt{C}_M]$ *cc-respecting* if, for all $a \in A$, we have $a \in P_{\wr U,R,m \wr}$ iff $|\mathtt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, U, R)| \geq m$.

Finally, we call $\mathfrak{A} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\} \cup \mathtt{C}_M]$ *well-typed* if it is eq-respecting and cc-respecting.

▶ **Example 10.** In Figure 2(c), where we suppose $M = 3$ and $\Sigma = \emptyset$, the element a satisfies the counting constraints $\wr \emptyset, \{(2,2)\}, 1 \wr$, $\wr \emptyset, \{(1,1),(2,2)\}, 1 \wr$, $\wr \emptyset, \{(1,2)\}, 2 \wr$, and $\wr \emptyset, \{(1,1),(1,2)\}, 3 \wr$, as well as all inherited constraints for smaller constants (which we omitted). We write $\wr \emptyset, R, m \wr$ as $R \geq m$. In fact, pairs from R are represented as black bars in the obvious way (cf. Figure 2(d)); moreover, for each constraint, the corresponding elements have the same color. Finally, the data structure from Figure 2(d) is well-typed, i.e., eq- and cc-respecting. Again, we omit inherited constraints. ◇

To summarize, we have the following reduction:

▶ **Lemma 11.** *For each formula $\varphi \in$ 1-Loc-FO$[\Sigma; \Gamma]$, we can effectively compute $M \in \mathbb{N}$ and $\chi \in \mathrm{FO}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M; \emptyset]$ such that φ is satisfiable iff χ has a well-typed model.*

Step 2: Well-Diagonalized Structures

In $\mathtt{C}_M$, we still have the diagonal relation $(1,2) \in \Gamma$. Our goal is to get rid of it so that we only deal with the diagonal-free set $\Gamma_{df} = \{(1,1),(2,2)\}$. The idea is again to extend a given structure $\mathfrak{A}$, but now we add new elements, one for each value $n \in \mathit{Val}_{\mathfrak{A}}$, which we tag with a unary symbol ed and whose two data values are n. Diagonal equality will be ensured through making a detour via these 'diagonal' elements (hence the name ed).

Formally, when we start from some $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\}]$, the data structure $\mathfrak{A} + \mathsf{ed} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}]$ is defined as $(A', f'_1, f'_2, (P'_\sigma))$ where $A' = A \uplus \mathit{Val}_{\mathfrak{A}}$, $f'_i(a) = f_i(a)$ for all $a \in A$ and $i \in \{1,2\}$, $f'_1(a) = f'_2(a) = a$ for all $a \in \mathit{Val}_{\mathfrak{A}}$, $P'_\sigma = P_\sigma$ for all $\sigma \in \Theta \setminus \{\mathsf{eq}\}$, $P'_{\mathsf{ed}} = \mathit{Val}_{\mathfrak{A}}$, and $P'_{\mathsf{eq}} = P_{\mathsf{eq}} \cup \mathit{Val}_{\mathfrak{A}}$.

▶ **Example 12.** The structure $\mathfrak{A} + \mathsf{ed}$ is illustrated in Figure 3(a), with $\Theta = \emptyset$. ◇

With this, we say that $\mathfrak{B} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}]$ is *well-diagonalized* if it is of the form $\mathfrak{A} + \mathsf{ed}$ for some *eq-respecting* $\mathfrak{A} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\}]$. Note that then $\mathfrak{B}$ is eq-respecting, too.

▶ **Example 13.** The data structure $\mathfrak{A} + \mathsf{ed}$ from Figure 3(a) is well-diagonalized. The one from Figure 3(b) is not well-diagonalized (in particular, it is not eq-respecting). ◇

We will need a way to ensure that the considered data structures are well-diagonalized. To this end, we introduce the following sentence from $\mathrm{FO}^2[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}; \Gamma_{df}]$:

$$\begin{aligned}\xi_{\mathsf{ed}}^{\Theta} := \quad & \textstyle\bigwedge_{i\in\{1,2\}} \forall x.\exists y.(\mathsf{ed}(y) \wedge x \;{}_i{\sim}_i\; y) \wedge \big(\forall x.\forall y.(\mathsf{ed}(x) \wedge \mathsf{ed}(y) \wedge x \;{}_i{\sim}_i\; y) \to x = y\big)\\ & \wedge \forall x.\mathsf{eq}(x) \leftrightarrow \exists y.(\mathsf{ed}(y) \wedge x \;{}_1{\sim}_1\; y \wedge x \;{}_2{\sim}_2\; y)\\ & \wedge \forall x.\mathsf{ed}(x) \to \textstyle\bigwedge_{\sigma\in\Theta} \neg\sigma(x)\end{aligned}$$

Every structure that is well-diagonalized satisfies $\xi_{\mathsf{ed}}^{\Theta}$. The converse is not true in general. In particular, a model of $\xi_{\mathsf{ed}}^{\Theta}$ is not necessarily eq-respecting. However, if a structure satisfies a formula $\varphi \in \mathrm{FO}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}; \Gamma_{df}]$, then it is possible to perform a permutation on the first (or the second) values of its elements while preserving φ. This allows us to get:

▶ **Lemma 14.** *Let $\mathfrak{B} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}]$ and $\varphi \in \mathrm{FO}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}; \Gamma_{df}]$. If $\mathfrak{B} \models \varphi \wedge \xi_{\mathsf{ed}}^{\Theta}$, then there exists an eq-respecting $\mathfrak{A} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\}]$ such that $\mathfrak{A} + \mathsf{ed} \models \varphi$.*

▶ **Example 15.** Consider Figure 3 and let $\Theta = \emptyset$. The data structure from Figure 3(b) satisfies $\xi_{\mathsf{ed}}^{\Theta}$, though it is not well-diagonalized. Suppose it also satisfies $\varphi \in \mathrm{FO}[\{\mathsf{eq}, \mathsf{ed}\}; \Gamma_{df}]$. By permutation of the first data values, we obtain the well-diagonalized data structure in Figure 3(a). As φ does not talk about the diagonal relation, satisfaction of φ is preserved. ◇

Finally, we can inductively translate $\varphi \in \mathrm{FO}[\Theta \cup \{\mathsf{eq}\}; \emptyset]$ into a formula $[\![\varphi]\!]_{+\mathsf{ed}} \in \mathrm{FO}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}; \emptyset]$ that avoids the extra 'diagonal' elements: $[\![\sigma(x)]\!]_{+\mathsf{ed}} = \sigma(x)$, $[\![x = y]\!]_{+\mathsf{ed}} = (x = y)$, $[\![\exists x.\varphi]\!]_{+\mathsf{ed}} = \exists x.(\neg\mathsf{ed}(x) \wedge [\![\varphi]\!]_{+\mathsf{ed}})$, $[\![\varphi \vee \varphi']\!]_{+\mathsf{ed}} = [\![\varphi]\!]_{+\mathsf{ed}} \vee [\![\varphi']\!]_{+\mathsf{ed}}$, and $[\![\neg\varphi]\!]_{+\mathsf{ed}} = \neg[\![\varphi]\!]_{+\mathsf{ed}}$. We immediately obtain:

▶ **Lemma 16.** *Let $\mathfrak{A} \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\}]$ and $\varphi \in \mathrm{FO}[\Theta \cup \{\mathsf{eq}\}; \emptyset]$ be a sentence. We have $\mathfrak{A} \models \varphi$ iff $\mathfrak{A} + \mathsf{ed} \models [\![\varphi]\!]_{+\mathsf{ed}}$.*

Step 3: Getting Rid Of the Diagonal Relation

We will now exploit well-diagonalized data structures to reason about environments relative to Γ in terms of environments relative to Γ_{df}. Recall that Θ ranges over finite sets such that $\Sigma \subseteq \Theta$.

▶ **Lemma 17.** *Let $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Theta \cup \{\mathsf{eq}\}]$ be eq-respecting and $\mathfrak{B} = \mathfrak{A} + \mathsf{ed}$. Moreover, let $a \in A$, $U \subseteq \Sigma$, and $R \subseteq \Gamma$ be a nonempty set. We have $\mathtt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, U, R) =$*

$$\begin{cases} \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \Gamma_{df}) \setminus P_{\mathsf{ed}} & \textit{if } a \in P_{\mathsf{eq}} \textit{ and } R = \Gamma & (1)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \Gamma_{df}) & \textit{if } a \notin P_{\mathsf{eq}} \textit{ and } R = \Gamma_{df} & (2)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \{(1,1)\}) \cap (P_{\mathsf{eq}} \setminus P_{\mathsf{ed}}) & \textit{if } a \notin P_{\mathsf{eq}} \textit{ and } R = \{(1,1),(1,2)\} & (3)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \{(2,2)\}) & \textit{if } a \in P_{\mathsf{eq}} \textit{ and } R = \{(2,2),(1,2)\} & (4)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \{(2,2)\}) \setminus P_{\mathsf{ed}} & \textit{if } a \notin P_{\mathsf{eq}} \textit{ and } R = \{(2,2)\} & (5)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, U, \{(1,1)\}) \setminus P_{\mathsf{eq}} & \textit{if} \qquad\quad R = \{(1,1)\} & (6)\\ \mathtt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(d, U, \{(2,2)\}) & \textit{if } a \notin P_{\mathsf{eq}} \textit{ and } R = \{(1,2)\} & (7)\\ \quad \textit{for the unique } d \in P_{\mathsf{ed}} \textit{ such that } d \;{}_1{\sim}_1^{\mathfrak{B}}\; a & & \\ \emptyset & \textit{otherwise} & (8) \end{cases}$$

▶ **Example 18.** Let us go through some cases of Lemma 17 using Figure 3(a), and letting $\Sigma = \Theta = \emptyset$.

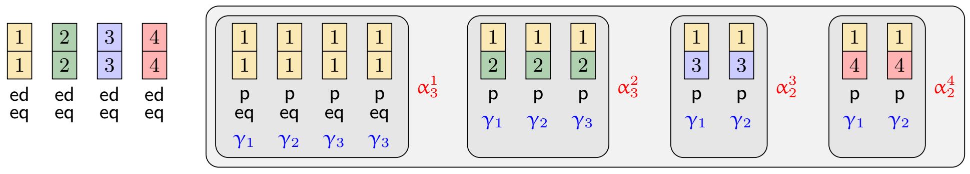

Figure 4 Counting intersections for $M = 3$ and elements with label p

(1) Let $a = a_1$ and $R = \Gamma$. Then, $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, R) = \{a_1, a_2, a_3\}$. We also have that $\texttt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, \emptyset, \Gamma_{df}) = \{a_1, a_2, a_3, b_1\}$: These are the elements that coincide with a *exactly* on the first and the on the second data value when we dismiss the diagonal relation. Of course, as we consider $\mathfrak{B}$, this includes b_1, which we have to exclude. Thus, $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, R) = \texttt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, \emptyset, \Gamma_{df}) \setminus P_{\mathsf{ed}}$.

(6) Let $a = a_4$ and $R = \{(1, 1)\}$. We have $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, R) = \{a_8\}$. Looking at $\mathfrak{B}$ and discarding the diagonal relation would also include b_3 and any element with data-value pair $(3, 3)$. Discarding P_{eq}, we obtain $\texttt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(a, \emptyset, \{(1, 1)\}) \setminus P_{\mathsf{eq}} = \{a_8, b_3\} \setminus \{b_3\} = \{a_8\}$.

(7) Let $a = a_7$ and $R = \{(1, 2)\}$. Then, $\texttt{Env}_{\mathfrak{A},\Sigma,\Gamma}(a, \emptyset, R) = \{a_4, a_5\}$, which is the set of elements whose second data value is 1 and whose first data value is different from 1. The idea is now to change the reference point. Take the unique $d \in P_{\mathsf{ed}}$ such that $d \;{}_1{\sim}_1^{\mathfrak{B}}\; a$. Thus, $d = b_1$. The set $\texttt{Env}_{\mathfrak{B},\Sigma,\Gamma_{df}}(b_1, \emptyset, \{(2, 2)\})$ gives us exactly the elements that have 1 as the second data value and a first value different from 1, as desired. ◇

Let us wrap up: By Lemmas 11 and 17, we end up with checking counting constraints in an extended data structure without using the diagonal relation.

Step 4: Counting in Two-Variable Logic

The next step is to express these constraints using two-variable formulas. Counting in two-variable logic is established using further unary predicates. These additional predicates allow us to define a partitioning of the universe of a structure into so-called *intersections*. Suppose $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \text{Data}[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\}]$, where $\Sigma \subseteq \Theta$. Let $a \in A \setminus P_{\mathsf{ed}}$ and define $\ell_\Sigma(a) = \{\sigma \in \Sigma \mid a \in P_\sigma\}$. The *intersection* of a in $\mathfrak{A}$ is the set $\{b \in A \setminus P_{\mathsf{ed}} \mid a \;{}_1{\sim}_1\; b \wedge a \;{}_2{\sim}_2\; b \wedge \ell_\Sigma(a) = \ell_\Sigma(b)\}$. A set is called an *intersection* in $\mathfrak{A}$ if it is the intersection of some $a \in A \setminus P_{\mathsf{ed}}$.

▶ **Example 19.** Consider Figure 4 and suppose $\Sigma = \{\mathsf{p}\}$. The intersections of the given data structure are gray-shaded. ◇

Let us introduce the various unary predicates, which will be assigned to *non-diagonal* elements. There are three types of them (for the first two types, also see Figure 4):

1. The unary predicates $\Lambda_M^\gamma = \{\gamma_1, \ldots, \gamma_M\}$ have the following intended meaning: For all intersections I and $i \in \{1, \ldots, M\}$, we have $|I| \geq i$ iff there is $a \in I$ such that $a \in P_{\gamma_i}$. In other words, the presence (or absence) of γ_i in an intersection I tells us whether $|I| \geq i$.
2. The predicates $\Lambda_M^\alpha = \{\alpha_i^j \mid i \in \{1, \ldots, M\}$ and $j \in \{1, \ldots, M + 2\}\}$ have the following meaning: If a is labeled with α_i^j, then (i) there are at least j intersections sharing the same first value and the same label set $\ell_\Sigma(a)$, and (ii) the intersection of a has i elements if $i \leq M - 1$ and at least M elements if $i = M$. Hence, in α_i^j, index i counts the elements inside an intersection, and j labels up to $M + 2$ different intersections. We need to go beyond M due to Lemma 17: When we remove certain elements (e.g., P_{eq}) from an environment, we must be sure to still have sufficiently many to be able to count until M.

3. Labels from $\Lambda_M^\beta = \{\beta_i^j \mid i \in \{1,\ldots,M\} \text{ and } j \in \{1,\ldots,M+1\}\}$ will play a similar role as those in Λ_M^α but consider the second values of the elements instead of the first ones.

▶ **Example 20.** A suitable labeling for types γ and α is illustrated in Figure 4 for $M = 3$. ◇

Let $\Lambda_M = \Lambda_M^\alpha \cup \Lambda_M^\beta \cup \Lambda_M^\gamma$ denote the set of all these unary predicates. It is relatively standard to come up with sentences $\varphi_\alpha, \varphi_\beta, \varphi_\gamma \in \mathrm{FO}^2[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\} \cup \Lambda_M; \Gamma_{df}]$ that guarantee the respective properties. In particular, they make use of the formula $x \;{}_1{\sim}_1\; y \wedge x \;{}_2{\sim}_2\; y \wedge \bigwedge_{\sigma\in\Sigma} \sigma(x) \leftrightarrow \sigma(y)$ saying that two (non-diagonal) elements x and y are in the same intersection.

Now that we can count on a consistent labeling with predicates from Λ_M, let us see how we can exploit it to express $\wr U, R, m \wr \in \mathtt{C}_M$, with additional help from Lemma 17, as a formula $\varphi_{U,R,m}(x) \in \mathrm{FO}^2[\Theta \cup \{\mathsf{eq}, \mathsf{ed}\} \cup \Lambda_M; \Gamma_{df}]$ applied to *non-diagonal* elements (outside P_{ed}). Let us look at two sample cases according to the case distinction done in Lemma 17. Hereby, we will use, for $U \subseteq \Sigma$, the formula $\varphi_U(y) = \bigwedge_{\sigma\in U} \sigma(y) \wedge \bigwedge_{\sigma\in\Sigma\setminus U} \neg\sigma(y)$.

(1) In this simple case with $R = \{(1,1),(2,2),(1,2)\}$, we need to say that (i) the element a under consideration is in P_{eq}, and (ii) there is an intersection of size at least m (i..e., it contains a γ_m-labeled element) whose elements b satisfy $a \;{}_1{\sim}_1\; b$, $a \;{}_2{\sim}_2\; b$, and $\ell_\Sigma(b) = U$:

$$\varphi_{U,R,m}(x) \;:=\; \mathsf{eq}(x) \wedge \exists y.\big(\varphi_U(y) \wedge x \;{}_1{\sim}_1\; y \wedge x \;{}_2{\sim}_2\; y \wedge \gamma_m(y)\big)$$

(6) For $R = \{(1,1)\}$, we first need an extra definition. Given $m \in \{1,\ldots,M\}$, we define the set $\mathcal{S}_{\alpha,m}$ of subsets of Λ_M^α as follows: $\mathcal{S}_{\alpha,m} = \{\{\alpha_{i_1}^{j_1},\ldots,\alpha_{i_k}^{j_k}\} \mid i_1+\ldots+i_k \geq m \text{ and } j_1 < j_2 < \ldots < j_k\}$. It corresponds to the sets of elements α_i^j whose sum of i is greater than or equal to m. We can then translate the constraint according to Lemma 17 as follows:

$$\varphi_{U,R,m}(x) \;:=\; \bigvee_{S\in\mathcal{S}_{\alpha,m}} \bigwedge_{\alpha\in S} \exists y.\big(\varphi_U(y) \wedge \alpha(y) \wedge \neg\mathsf{eq}(y) \wedge x \;{}_1{\sim}_1\; y \wedge \neg(x \;{}_2{\sim}_2\; y)\big)$$

Finally, it remains to say that all elements are labeled with the suitable counting constraints. So we let $\varphi_{cc} = \forall x.\neg\mathsf{ed}(x) \rightarrow \bigwedge_{\wr U,R,m\wr \in \mathtt{C}_M} \wr U, R, m\wr(x) \leftrightarrow \varphi_{U,R,m}(x)$.

▶ **Lemma 21.** *Let $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M \cup \Lambda_M]$ be eq-respecting. If $\mathfrak{A} + \mathsf{ed} \models \varphi_\alpha \wedge \varphi_\beta \wedge \varphi_\gamma \wedge \varphi_{cc}$, then $\mathfrak{A}$ is cc-respecting.*

Step 5: Putting it All Together

Let $\mathsf{All} = \Sigma \cup \{\mathsf{eq}, \mathsf{ed}\} \cup \mathtt{C}_M \cup \Lambda_M$ denote the set of all the unary predicates that we have introduced so far. Recall that, after Step 1, we were left with $M \geq 1$ and a formula $\varphi \in \mathrm{FO}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M; \emptyset]$. The question is now whether φ has a well-typed model (i.e., a model that is eq-respecting and cc-respecting). Altogether, we get the following reduction:

▶ **Proposition 22.** *Let $\varphi \in \mathrm{FO}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M; \emptyset]$. Then, φ has a well-typed model iff $\widehat{\varphi} := [\![\varphi]\!]_{+\mathsf{ed}} \wedge \xi_{\mathsf{ed}}^{\mathsf{All}\setminus\{\mathsf{eq},\mathsf{ed}\}} \wedge \varphi_\alpha \wedge \varphi_\beta \wedge \varphi_\gamma \wedge \varphi_{cc} \in \text{ext-}\mathrm{FO}^2[\mathsf{All}; \Gamma_{df}]$ is satisfiable.*

Proof. Suppose $\widehat{\varphi}$ is satisfiable. Then, there is $\mathfrak{B} \in \mathrm{Data}[\mathsf{All}]$ such that $\mathfrak{B} \models \widehat{\varphi}$. By Lemma 14, there exists an eq-respecting data structure $\mathfrak{A} \in \mathrm{Data}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M \cup \Lambda_M]$ such that $\mathfrak{A} + \mathsf{ed} \models [\![\varphi]\!]_{+\mathsf{ed}} \wedge \varphi_\alpha \wedge \varphi_\beta \wedge \varphi_\gamma \wedge \varphi_{cc}$. Using Lemma 21, we deduce that $\mathfrak{A}$ is cc-respecting and, thus, well-typed. Furthermore, by Lemma 16, we have $\mathfrak{A} \models \varphi$. Note that $\mathfrak{A}$ belongs to $\mathrm{Data}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M \cup \Lambda_M]$. However, by removing the unary predicates in Λ_M, we still have a model of φ from $\mathrm{Data}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M]$ as required. Hence, φ has a well-typed model.

Assume now that there exists a well-typed data structure $\mathfrak{A} \in \mathrm{Data}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M]$ such that $\mathfrak{A} \models \varphi$. Using Lemma 16, we have that $\mathfrak{A} + \mathsf{ed} \models [\![\varphi]\!]_{+\mathsf{ed}}$. Furthermore, using the fact that $\mathfrak{A}$ is well-typed, we can add the unary predicates from Λ_M to $\mathfrak{A} + \mathsf{ed}$ to obtain a data structure $\mathfrak{A}'$ in Data[All] such that $\mathfrak{A}' \models \varphi_\alpha \wedge \varphi_\beta \wedge \varphi_\gamma \wedge \varphi_{cc}$. Note that $\mathfrak{A}'$ is well-diagonalized. We deduce that $\mathfrak{A}' \models \widehat{\varphi}$. ◀

▶ **Theorem 23.** $\mathrm{DataSat}(\text{1-Loc-FO}, \{(1,1),(2,2),(1,2)\})$ *is decidable.*

Proof. Let $\psi \in \text{1-Loc-FO}[\Sigma; (1,1),(2,2),(1,2)]$. Using Lemma 11, we can effectively compute $M \in \mathbb{N}$ and $\varphi \in \mathrm{FO}[\Sigma \cup \{\mathsf{eq}\} \cup \mathtt{C}_M; \emptyset]$ such that ψ is satisfiable iff φ has a well-typed model. By Proposition 22, φ has a well-typed model iff $\widehat{\varphi}$ is satisfiable. Since $\widehat{\varphi}$ belongs to $\text{ext-FO}^2[\mathsf{All}; \Gamma_{df}]$, we conclude using Proposition 6. ◀

4 Undecidability Results

Let us show that extending the neighborhood radius yields undecidability. We rely on a reduction from the domino problem [9] and use a specific technique presented in [25].

The Tiling Problem. A *domino system* $\mathcal{D}$ is a triple (D, H, V) where D is a finite set of dominoes and $H, V \subseteq D \times D$ are two binary relations. Let $\mathfrak{G}_m$ denote the standard grid on an $m \times m$ torus, i.e., $\mathfrak{G}_m = (G_m, H_m, V_m)$ where H_m and V_m are two binary relations defined as follows: $G_m = \mathbb{Z} \bmod m \times \mathbb{Z} \bmod m$, $H_m = \{((i,j),(i',j)) \mid i' - i \equiv 1 \mod m\}$, and $V_m = \{((i,j),(i,j')) \mid i' - i \equiv 1 \mod m\}$. In the sequel, we will suppose $\mathbb{Z} \bmod m = \{0, \ldots, m-1\}$ using the least positive member to represent residue classes.

A *bi-binary structure* is a triple (A, R_1, R_2) where A is a finite set and R_1, R_2 are subsets of $A \times A$. Domino systems and $\mathfrak{G}_m$ for any m are examples of bi-binary structures. For two bi-binary structures $\mathfrak{G} = (G, H, V)$ and $\mathfrak{G}' = (G', H', V')$, we say that $\mathfrak{G}$ is *homomorphically embeddable* into $\mathfrak{G}'$ if there is a morphism $\pi : \mathfrak{G} \to \mathfrak{G}'$, i.e., a mapping π such that, for all $a, a' \in G$, $(a, a') \in H \Rightarrow (\pi(a), \pi(a')) \in H'$ and $(a, a') \in V \Rightarrow (\pi(a), \pi(a')) \in V'$. For instance, $\mathfrak{G}_{k \cdot m}$ is homomorphically embeddable into $\mathfrak{G}_m$ through reduction mod m. For a domino system $\mathcal{D}$, a *periodic tiling* is a morphism $\tau : \mathfrak{G}_m \to \mathcal{D}$ for some m and we say that $\mathcal{D}$ *admits a periodic tiling* if there exists a periodic tiling of $\mathcal{D}$.

The problem Tiles (or *periodic tiling problem*), which is well known to be undecidable [9], is defined as follows: Given a domino system $\mathcal{D}$, does $\mathcal{D}$ admit a periodic tiling?

To use Tiles in our reductions, we first use some specific bi-binary structures, which we call grid-like and which are easier to manipulate in our context to encode domino systems. A bi-binary structure $\mathfrak{G} = (A, H, V)$ is said to be *grid-like* if some $\mathfrak{G}_m$ is homomorphically embeddable into $\mathfrak{G}$. The logic FO *over bi-binary structures* refers to the first-order logic on two binary relations H, V, and we write $\mathsf{H}xy$ to say that x and y are in relation for H. Consider the two following FO formulas over bi-binary structures: $\varphi_{complete} = \forall x.\forall y.\forall x'.\forall y'.((\mathsf{H}xy \wedge \mathsf{V}xx' \wedge \mathsf{V}yy') \to \mathsf{H}x'y')$ and $\varphi_{progress} = \forall x.(\exists y.\mathsf{H}xy \wedge \exists y.\mathsf{V}xy)$. The following lemma, first stated and proved in [25], shows that these formulas suffice to characterize grid-like structures:

▶ **Lemma 24** ([25]). *Let $\mathfrak{G} = (A, H, V)$ be a bi-binary structure. If $\mathfrak{G}$ satisfies $\varphi_{complete}$ and $\varphi_{progress}$, then $\mathfrak{G}$ is grid-like.*

Given $\mathfrak{A} = (A, f_1, f_2, (P_\sigma)) \in \mathrm{Data}[\Sigma]$ and $\varphi(x,y) \in \mathrm{FO}[\Sigma; \Gamma]$, we define the binary relation $[\![\varphi]\!]_{\mathfrak{A}} = \{(a,b) \in A \times A \mid \mathfrak{A} \models_{I[x/a][y/b]} \varphi(x,y)$ for some interpretation function $I\}$.

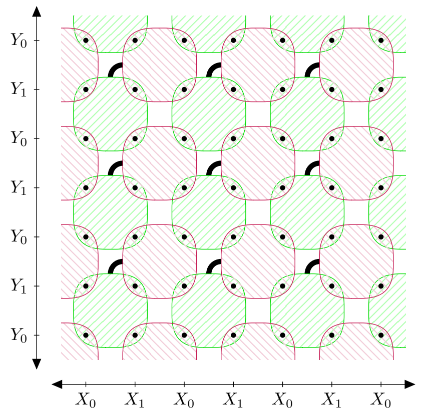

Figure 5 The local pattern of $\mathfrak{A}_{2m}$. Dots denote elements. Two dots are in the same ${}_1{\sim}_1$-equivalence class (resp. ${}_2{\sim}_2$) iff they are in the same green (resp. purple) area. The thick black lines represent the relation ${}_1{\sim}_2$ in the following way: if a ${}_1{\sim}_1$-equivalence class C_1 and a ${}_2{\sim}_2$-equivalence class C_2 are connected with a thick black line, then for any $a \in C_1$ and $b \in C_2$, we have $a \;{}_1{\sim}_2\; b$.

$$\begin{aligned}
\varphi_H^{00} &= X_0(x) \wedge X_1(y) \wedge Y_0(x) \wedge Y_0(y) \wedge x \;{}_1{\sim}_1\; y &\quad \varphi_V^{00} &= X_0(x) \wedge X_0(y) \wedge Y_0(x) \wedge Y_1(y) \wedge x \;{}_1{\sim}_1\; y\\
\varphi_H^{10} &= X_1(x) \wedge X_0(y) \wedge Y_0(x) \wedge Y_0(y) \wedge x \;{}_2{\sim}_2\; y &\quad \varphi_V^{10} &= X_1(x) \wedge X_1(y) \wedge Y_0(x) \wedge Y_1(y) \wedge x \;{}_1{\sim}_1\; y\\
\varphi_H^{01} &= X_0(x) \wedge X_1(y) \wedge Y_1(x) \wedge Y_1(y) \wedge x \;{}_1{\sim}_1\; y &\quad \varphi_V^{01} &= X_0(x) \wedge X_0(y) \wedge Y_1(x) \wedge Y_0(y) \wedge x \;{}_2{\sim}_2\; y\\
\varphi_H^{11} &= X_1(x) \wedge X_0(y) \wedge Y_1(x) \wedge Y_1(y) \wedge x \;{}_2{\sim}_2\; y &\quad \varphi_V^{11} &= X_1(x) \wedge X_1(y) \wedge Y_1(x) \wedge Y_0(y) \wedge x \;{}_2{\sim}_2\; y\\
\varphi_H &= \varphi_H^{00} \vee \varphi_H^{10} \vee \varphi_H^{01} \vee \varphi_H^{11} &\quad \varphi_V &= \varphi_V^{00} \vee \varphi_V^{10} \vee \varphi_V^{01} \vee \varphi_V^{11}
\end{aligned}$$

Figure 6 Link between $\mathfrak{A}_{2m}$ and $\mathfrak{G}_{2m}$

Thus, given two FO$[\Sigma;\Gamma]$ formulas $\varphi_1(x,y), \varphi_2(x,y)$ with two free variables, $(A, [\![\varphi_1]\!]_{\mathfrak{A}}, [\![\varphi_2]\!]_{\mathfrak{A}})$ is a bi-binary structure.

As we want to reason on data structures, we build a data structure $\mathfrak{A}_{2m}$ that corresponds to the grid $\mathfrak{G}_{2m} = (G_{2m}, H_{2m}, V_{2m})$. This structure is depicted locally in Figure 5. To define $\mathfrak{A}_{2m}$, we use four unary predicates given by $\Sigma_{grid} = \{X_0, X_1, Y_0, Y_1\}$. They give us access to the coordinate modulo 2. We then define $\mathfrak{A}_{2m} = (G_{2m}, f_1, f_2, (P_\sigma)) \in \text{Data}[\Sigma_{grid}]$ as follows: For $k \in \{0,1\}$, we have $P_{X_k} = \{(i,j) \in G_{2m} \mid i \equiv k \mod 2\}$ and $P_{Y_k} = \{(i,j) \in G_{2m} \mid j \equiv k \mod 2\}$. For all $i,j \in \{0,\dots,2m-1\}$, we set $f_1(i,j) = ((i/2) \mod m) + m * ((j/2) \mod m)$ (where $/$ stands for the Euclidian division). Finally, for all $i,j \in \{1,\dots,2m\}$, set $f_2(i \mod (2m), j \mod (2m)) = f_1(i-1, j-1)$.

In Figure 6, we define quantifier free formulas $\varphi_H(x,y)$ and $\varphi_V(x,y)$ from the logic FO$[\Sigma_{grid}; (1,1),(2,2)]$ with two free variable. These formulas allow us to make the link between the data structure $\mathfrak{A}_{2m}$ and the grid $\mathfrak{G}_{2m}$, and we will use them later on to ensure that a data structure has a shape 'similar' to $\mathfrak{A}_{2m}$.

▶ Remark 25. Note that, using the definitions of G_{2m} and of $\mathfrak{A}_{2m}$ we can show that, if $\mathfrak{G}$ is the bi-binary structure $(G_{2m}, [\![\varphi_H]\!]_{\mathfrak{A}_{2m}}, [\![\varphi_V]\!]_{\mathfrak{A}_{2m}})$, then $\mathfrak{G}_{2m} = \mathfrak{G}$.

The Reduction from Radius 3. We first use the previously introduced notions to show that DataSat(3-Loc-FO, $\{(1,1),(2,2)\}$) is undecidable, hence we assume now that $\Gamma = \{(1,1),(2,2)\}$. The first step in our reduction from TILES consists in defining $\varphi_{grid}^{3\text{-}loc} \in$ 3-Loc-FO$[\Sigma_{grid}; (1,1),(2,2)]$ to check that a data structure corresponds to a grid ($\oplus$ stands for exclusive or):

$$\begin{aligned}
\varphi_{complete}^{3\text{-}loc} &= \forall x. \langle\!\langle \forall y. \forall x'. \forall y'. \varphi_H(x,y) \wedge \varphi_V(x,x') \wedge \varphi_V(y,y') \to \varphi_H(x',y') \rangle\!\rangle_x^3\\
\varphi_{progress}^{3\text{-}loc} &= \forall x. \langle\!\langle \exists y. \varphi_H(x,y) \wedge \exists y. \varphi_V(x,y) \rangle\!\rangle_x^3\\
\varphi_{grid}^{3\text{-}loc} &= \varphi_{complete}^{3\text{-}loc} \wedge \varphi_{progress}^{3\text{-}loc} \wedge \forall x. \langle\!\langle (X_0(x) \oplus X_1(x)) \wedge (Y_0(x) \oplus Y_1(x)) \rangle\!\rangle_x^3
\end{aligned}$$

▶ **Lemma 26.** *We have $\mathfrak{A}_{2m} \models \varphi^{3\text{-}loc}_{grid}$. Moreover, for all $\mathfrak{A} = (A, f_1, f_2, (P_\sigma))$ in $\mathrm{Data}[\Sigma_{grid}]$, if $\mathfrak{A} \models \varphi^{3\text{-}loc}_{grid}$, then $(A, [\![\varphi_H]\!]_{\mathfrak{A}}, [\![\varphi_V]\!]_{\mathfrak{A}})$ is grid-like.*

Given a domino system $\mathcal{D} = (D, H_\mathcal{D}, V_\mathcal{D})$, we now provide a formula $\varphi_\mathcal{D}$ from the logic 3-Loc-FO$[D; (1,1), (2,2)]$ that guarantees that, if a data structure corresponding to a grid satisfies $\varphi_\mathcal{D}$, then it can be embedded into $\mathcal{D}$:

$$\begin{aligned}\varphi_\mathcal{D} \quad := \quad & \forall x.\langle\!\langle \bigvee_{d \in D} \Big(d(x) \wedge \bigwedge_{d \neq d' \in D} \neg(d(x) \wedge d'(x)) \Big) \rangle\!\rangle^3_x \\ & \wedge \forall x.\langle\!\langle \forall y.\varphi_H(x,y) \rightarrow \bigvee_{(d,d') \in H_\mathcal{D}} d(x) \wedge d'(y) \rangle\!\rangle^3_x \\ & \wedge \forall x.\langle\!\langle \forall y.\varphi_V(x,y) \rightarrow \bigvee_{(d,d') \in V_\mathcal{D}} d(x) \wedge d'(y) \rangle\!\rangle^3_x\end{aligned}$$

▶ **Proposition 27.** *Given $\mathcal{D} = (D, H_\mathcal{D}, V_\mathcal{D})$ a domino system, $\mathcal{D}$ admits a periodic tiling iff the* 3-Loc-FO$[\Sigma_{grid} \uplus D; (1,1), (2,2)]$ *formula $\varphi^{3\text{-}loc}_{grid} \wedge \varphi_\mathcal{D}$ is satisfiable.*

As a corollary of the proposition, we obtain the main result of this section.

▶ **Theorem 28.** DataSat(3-Loc-FO, $\{(1,1),(2,2)\}$) *is undecidable.*

We can also reduce Tiles to DataSat(2-Loc-FO, $\{(1,1),(2,2),(1,2)\}$). In that case, it is a bit more subtle to build a formula similar to the formula $\varphi_{complete}$ as we have only neighborhood of radius 2, but we use the diagonal binary relation $(1,2)$ to overcome this.

▶ **Theorem 29.** DataSat(2-Loc-FO, $\{(1,1),(2,2),(1,2)\}$) *is undecidable.*

5 Future Work

There are some interesting open questions. For example, we leave open whether our main decidability result holds for two diagonal relations. Recall that, when comparing the expressiveness, two-variable first-order logic can be embedded in our logic. We do not know yet whether the converse holds. Until now our work has focused on the satisfiability problem. Another next step would be to see how our logic can be used to verify practical distributed algorithms.

References

1 C. Aiswarya, B. Bollig, and P. Gastin. An automata-theoretic approach to the verification of distributed algorithms. *Inf. Comput.*, 259(Part 3):305–327, 2018.

2 B. Bednarczyk and P. Witkowski. A Note on C^2 Interpreted over Finite Data-Words. In *27th International Symposium on Temporal Representation and Reasoning, TIME 2020, September 23-25, 2020, Bozen-Bolzano, Italy*, volume 178 of *LIPIcs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.TIME.2020.17.

3 H. Björklund and M. Bojanczyk. Shuffle expressions and words with nested data. In Ludek Kucera and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 750–761. Springer, 2007.

4 R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.

5 M. Bojanczyk, C. David, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data words. *ACM Trans. Comput. Log.*, 12(4):27:1–27:26, 2011.

6 M. Bojanczyk, A. Muscholl, T. Schwentick, and L. Segoufin. Two-variable logic on data trees and XML reasoning. *J. ACM*, 56(3), 2009.

7 B. Bollig, P. Bouyer, and F. Reiter. Identifiers in registers - describing network algorithms with logic. In *FOSSACS'19*, volume 11425 of *LNCS*, pages 115–132. Springer, 2019.

8 B. Bollig and D. Kuske. An optimal construction of hanf sentences. *J. Appl. Log.*, 10(2):179–186, 2012. doi:10.1016/j.jal.2012.01.002.

9 E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem.* Perspectives in Mathematical Logic. Springer, 1997.

10 N. Decker, P. Habermehl, M. Leucker, and D. Thoma. Ordered navigation on multi-attributed data words. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2014.

11 E. A. Emerson and K. S. Namjoshi. On reasoning about rings. *Int. J. Found. Comput. Sci.*, 14(4):527–550, 2003.

12 J. Esparza. Keeping a crowd safe: On the complexity of parameterized verification (invited talk). In *STACS'14)*, volume 25 of *LIPIcs*, pages 1–10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

13 W. Fokkink. *Distributed Algorithms: An Intuitive Approach.* MIT Press, 2013.

14 S. Grumbach and Z. Wu. Logical locality entails frugal distributed computation over graphs (extended abstract). In *WG'09*, volume 5911 of *LNCS*, pages 154–165. Springer, 2009.

15 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*, pages 132–145. North Holland, 1965.

16 A. Janiczak. Undecidability of some simple formalized theories. *Fundamenta Mathematicae*, 40:131–139, 1953.

17 A. Kara, T. Schwentick, and T. Zeume. Temporal logics on words with multiple data values. In Kamal Lodaya and Meena Mahajan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, volume 8 of *LIPIcs*, pages 481–492. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

18 E. Kieronski. Results on the guarded fragment with equivalence or transitive relations. In C.-H. Luke Ong, editor, *CSL'05*, volume 3634 of *LNCS*, pages 309–324. Springer, 2005.

19 E. Kieronski, J. Michaliszyn, I. Pratt-Hartmann, and L. Tendera. Two-variable first-order logic with equivalence closure. In *LICS'12*, pages 431–440. IEEE, 2012.

20 E. Kieronski and L. Tendera. On finite satisfiability of two-variable first-order logic with equivalence relations. In *LICS'09*, pages 123–132. IEEE, 2009.

21 I. V. Konnov, H. Veith, and J. Widder. What you always wanted to know about model checking of fault-tolerant distributed algorithms. In *PSI'15 in Memory of Helmut Veith*, volume 9609 of *LNCS*, pages 6–21. Springer, 2015.

22 L. Libkin. *Elements of Finite Model Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

23 N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers Inc., 1996.

24 A. Manuel and T. Zeume. Two-variable logic on 2-dimensional structures. In Simona Ronchi Della Rocca, editor, *Computer Science Logic 2013 (CSL 2013), CSL 2013, September 2-5, 2013, Torino, Italy*, volume 23 of *LIPIcs*, pages 484–499. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.

25 M. Otto. Two-variable first-order logic over ordered domains. *Journal of Symbolic Logic*, 66:685–702, 2001.

26 L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *CSL'06*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.

27 T. Tan. Extending two-variable logic on data trees with order on data values and its automata. *ACM Trans. Comput. Log.*, 15(1):8:1–8:39, 2014.