Δ: Set-Theoretic Query Language Capturing LOGSPACE *

Alexander Leontjev [a] Vladimir Sazonov [b,**]

[a] *Program Systems Institute of Russian Academy of Sciences, Pereslavl-Zalessky, 152140, Russia*

[b] *Department of Computer Science, University of Liverpool, Liverpool L69 7ZF, U.K.*

E-mail: V.Sazonov@csc.liv.ac.uk

Two versions of a set theoretic Δ-language are considered as theoretical prototypes for database query languages with "nested", "unstructured" data where database states and queries are represented as hereditarily-finite (**HF**) sets and set theoretic operations, respectively. It is shown that these versions correspond exactly to (N/D)LOGSPACE computability over **HF** relative to a special graph representation (encoding) of **HF**-sets. Such languages over sets, capturing also PTIME, were introduced in previous works, however, descriptions of LOGSPACE over **HF** [34] were not completely satisfactory. Here we overcome the drawbacks of the previous approaches by some new partial result on definability of a linear ordering over finite extensional acyclic graphs and present a unified and simplified approach.

Keywords: finite set theory, finite model theory, descriptive complexity, query languages, semi-structured, nested, complex databases, capturing LOGSPACE, PTIME, transitive closure, definability

AMS Subject classification: Primary 68P15 Database theory; Secondary 03E47 Set-theoretic definability, 03E75 Applications of set theory.

1. Introduction

Nowadays, it has been established as a natural approach to characterise expressive power of database query languages by logics over finite structures or in terms of computational complexity. Vice versa, given complexity classes such as PTIME or LOGSPACE, we could ask whether there exist natural query languages corresponding to these classes. Thus, database theory is related to descriptive complexity theory and finite model theory (see, for example, [4]). Actually, it seems natural that database querying is related with some restricted notions of

* Extended version of [32].

** Corresponding author, working on various versions of this paper also in Program Systems Institute of Russian Academy of Sciences, Pereslavl-Zalessky (Russia), Siegen University (FRG) and Manchester Metropolitan University (UK).

computability: retrieving data from a database is something more special than arbitrary computation.

Following this general approach, and also continuing and strengthening the previous work [34] (see also Sect. 2), we consider in this paper two versions of a set theoretic Δ-language as theoretical prototypes for database query languages with "nested" data where *database states* and *queries* are interpreted, respectively, as *hereditarily-finite (***HF***) sets* and *set theoretic operations*. It is shown that these versions correspond exactly to (N/D)LOGSPACE computability over **HF**, respectively. This is done by using some new partial result on definability of a linear ordering over finite extensional acyclic graphs (Proposition 5.1). A well-known result of N. Immerman [26] on capturing (N/D)LOGSPACE in extensions of first-order logic by (nondeterministic/deterministic) transitive closure operators is also used, as well as a result of N. Immerman and R. Szelepcsényi [27,52] that Co-NLOGSPACE=NLOGSPACE which makes the logical description of NLOGSPACE more natural (the language is closed under negation).

To deal with computability over **HF**-sets we assume in this paper that (a) **HF**-sets, (b) membership relation between sets and (c) computable set operations are naturally represented, respectively, by (a′) vertices of acyclic graphs, (b′) the graph edges and (c′) corresponding graph transformers. See [42,44–49,34–36] for a more extensive general introduction to the subject and related results. The next two sections give a more technical introduction and some preliminaries.

Actually, we consider this set-theoretic approach to databases as a *straightforward generalisation of the well-known relational approach.* Instead of relations (sets of tuples) we are working directly with sets (of sets of sets, etc.) of data. On the other hand, as the structure of **HF**-sets is, in principle, very arbitrary, unlike the relational case, this is, simultaneously, a natural approach to *unstructured databases*. Unlike many others, but analogously to [14–16] we deal with *untyped* and *cumulative hierarchy* of **HF**-sets. There are also direct connections with the traditional descriptive complexity theory and finite model theory.

We discuss in Sect. 2 the previous approach to LOGSPACE over **HF**. Preliminaries on a set-theoretic approach to databases, queries and computability are presented in Sect. 3. We describe Δ-languages of set theoretic operations (queries) in Sect. 4. The Main Result on capturing NLOGSPACE and DLOGSPACE over **HF** is presented and proved in Sect. 5. Finally, we end with some concluding remarks and open questions in Sect. 6.

Note, that this paper is mostly self-contained, except some clearly isolated parts, but the exposition is sometimes rather condensed because, otherwise, the volume would be too large. Sometimes we also deviate from the primary technical goal of the paper (of capturing (N/D)LOGSPACE over **HF**) to present a wider database context and to make our general set-theoretic approach to databases more clear. We will mark such places by the word **'digression'** or use footnotes. However, many footnotes are quite essential for what follows.

2. Notes on a previous approach to LOGSPACE over HF

Some languages over hereditarily-finite sets, capturing also PTIME, were introduced in previous work [42,44–49,34–36]. However the case of LOGSPACE could not have been covered completely satisfactory in [34]. Two approaches to defining and capturing LOGSPACE computability over **HF** were presented (cf. formulas (10) and (12) in [34]) which were assumed as being essentially different at the time of writing of [34]. In one of them there were *problems with proving the closure under composition of queries*, whereas LOGSPACE was described in a sense exactly. In another one the problems with composition of queries were resolved by *some complication of the notion of LOGSPACE computability over* **HF** (anyway, there is no unique, canonical or most preferable definition for the case of **HF**). *The syntax of the corresponding query language appears rather unnatural* in comparison with other fundamental set theoretic operations, e.g. union or intersection. That is, it requires an infinite family of constant symbols. Nevertheless, it remains formally effective (computable; see Def. 3.3 below).

In this paper it hardly makes sense to describe these two approaches. Especially the second one is somewhat over-complicated — a price payed for the achieved closure under composition, as mentioned above. At the time of writing [34] this was a necessary intermediate step in a direction where a reasonable solution would be found. In a sense it was really the right way of thought leading to this present paper. Here we overcome the drawbacks of both approaches by presenting a *unified and simplified approach*. Indeed, all of them prove to be *equivalent* due to some new partial result, Proposition 5.1, on definability of a linear ordering over finite extensional acyclic graphs. (Cf. also Footnote 21 below.) In particular, we will obtain a natural and exact description of LOGSPACE over **HF** with the necessary closure properties. Note, that the second approach of [34] did not aim to have a result like Proposition 5.1 which remained as an open question. (Cf. a weaker question 6 in Sect. 13 of [34].)

3. Preliminaries and technical introduction

3.1. Hereditarily-finite sets as database states

Definition 3.1. The universe **HF** of "pure" *hereditarily-finite sets*, considered here also as abstract, *pure database states*, is defined inductively as the least class of sets such that

- if $x_1, \ldots, x_n \in \mathbf{HF}$, $n \geq 0$, then $\{x_1, \ldots, x_n\} \in \mathbf{HF}$ (with the order and duplication of x_i considered as irrelevant).

In particular, we have the *empty* set $\{\} = \emptyset \in \mathbf{HF}$ (the case of $n = 0$) and also $\{\emptyset\}$, $\{\{\emptyset\}\}$, $\{\emptyset, \{\emptyset\}\}$, etc. $\in \mathbf{HF}$.

3.1.1. **Digression:** *Attributes and urelements*

For real database applications we have to consider, as in [45,46] (and in [36] for hyper-sets), a more general universe $\mathbf{HF}(\mathcal{A},\mathcal{U})$ with *urelements* (or *atoms, elementary data*) $\mathcal{U}$ and database *attributes* (or *labels*) $\mathcal{A}$ defining it inductively by:

- $\mathcal{U} \subseteq \mathbf{HF}(\mathcal{A},\mathcal{U})$ and
- if $x_1, \ldots, x_n \in \mathbf{HF}(\mathcal{A},\mathcal{U})$ and $A_1, \ldots, A_n \in \mathcal{A}$, $n \geq 0$, then $\{A_1 : x_1, \ldots, A_n : x_n\} \in \mathbf{HF}(\mathcal{A},\mathcal{U})$ where $A_i : x_i$ are elements x_i labeled by the attributes $A_i \in \mathcal{A}$.

We may take $\mathcal{A} = \mathcal{U} =$ a set of finite strings in an alphabet, or we may include in $\mathcal{A} = \mathcal{U}$ also integers and other datatypes. Also, we could take $\mathcal{U} = \emptyset$ in favor of $\mathcal{A}$, essentially without any loss of generality. Thus, attributes may play the role of elementary data (urelements).

3.1.2. **Digression:** *Database states with on attributes and urelements*

As an example of data represented by such a hereditarily finite set consider

```
{ STUDENT:I.Ivanov, BIRTH_YEAR:1981, DEPT:CS,
  STUDIES:Logic, STUDIES:Data_Base_Theory }
```

with `STUDENT`, `BIRTH_YEAR`, `DEPT`, `STUDIES` $\in \mathcal{A}$ and `I.Ivanov`, `1981`, `CS`, etc. $\in \mathcal{U}$. (If it is considered that $\mathcal{U} = \emptyset$ then, instead of urelements such as `I.Ivanov`, we could consider singletons like `{I.Ivanov:{}}` with `I.Ivanov` $\in \mathcal{A}$ being now an attribute or a label of the empty set `{}`, and analogously for `1981` and other urelements.) Instead of considering `CS` (Computer Science Department) as an urelement we could take it as (the name of) a complicated data (set) of analogous form

```
CS = { ADMIN:..., LECTURERS:..., STUDENTS:... },
```

etc., with any deep nesting, if necessary. This way *nested* or *complex* data [2,12,14–16,20,28,31] may be represented as hereditarily-finite sets with attributes and urelements. Note, that any ordinary *relational database* may be presented by a hereditarily-finite set of the *nesting depth* $= 3$ ($= 4$, if there are only attributes and no urelements). Thus, a relational database is a finite *set* of relations $\{R : Rel, R' : Rel', \ldots, R^{(m)} : Rel^{(m)}\}$ named/labeled by $R, R', \ldots, R^{(m)}$ and represented visually by tables

$$Rel : \quad \begin{array}{|c|c|c|c|}\hline \multicolumn{4}{|c|}{R} \\ \hline A_1 & A_2 & \ldots & A_n \\ \hline\hline x_1 & x_2 & \ldots & x_n \\ \hline y_1 & y_2 & \ldots & y_n \\ \hline \vdots & \vdots & \ldots & \vdots \\ \hline \end{array} \tag{3.1}$$

Further, each relation is a finite *set* of tuples (records) i.e. of the "plain" *sets* $\{A_1 : x_1, \ldots, A_n : x_n\}$, etc.:

$$Rel = \{\text{tuple} : \{A_1 : x_1, \ldots, A_n : x_n\}, \text{tuple} : \{A_1 : y_1, \ldots, A_n : y_n\}, \ldots\}$$

with different attributes A_i and urelements $x_i, y_i, \ldots$ as corresponding values.

In principle, an arbitrary hereditarily-finite set s in $\mathbf{HF}(\mathcal{A},\mathcal{U})$ may be considered as a *database state*. Here the picture is more general than in ordinary relational databases. In this sense any unique relation or even any tuple or, trivially, any urelement may serve as a database state.

More general, but quite analogous and even more natural (not so unnecessarily restrictive), are cyclic, *semistructured* or *Web-like, hyper-linked* data [1,6,30,31,39]. See especially papers [9,36] based on the notion of bisimulation of graphs and therefore having also essentially *hyper-set-theoretic nature.* For example, `CS` above might even refer again to the first displayed set corresponding to a concrete student `I.Ivanov`. However, this paper, except several separate notes, does not account "officially" such possibilities for cycling. Cf. also Footnote 5 below. It is a separate question how to extend the results of this paper to hyper-sets.

3.1.3. **Digression:** *On database schemas*

Tuples in any relation (3.1) have a fixed number n of elements. Moreover, in each column of the table elements $x_i, y_i, \ldots$ are required to be taken from the same domain $D_{A_i} \subseteq \mathcal{U}$ "described" by the corresponding attribute A_i. In this sense, $\{A_1, \ldots, A_n\}$ usually defines a schema of the relation Rel, and a set of schemas of relations

$$\{R : \{A_1, \ldots, A_n\}, R' : \{A'_1, \ldots, A'_{n'}\}, \ldots, R^{(m)} : \{A_1^{(m)}, \ldots, A_{n^{(m)}}^{(m)}\}\}$$

defines a schema of the relational database.

In the case of a general database state s as an arbitrary set in $\mathbf{HF}(\mathcal{A},\mathcal{U})$ we also can appropriately define some weaker notion of *database schema* which describes a general form of the state s. We refer the reader to [46] and to, actually, a direct generalisation of that approach to the case of Web-like (or semistructured) databases [36,10] with cycles allowed. However, the query language Δ and its versions considered in Sect. 4 do not assume that any database schema is fixed. It is really a *general* language for *unstructured* **HF**-set data. The only structure which an **HF**-set carries is that it is a set of sets of sets, etc., with an arbitrary finite depth. It is usually said also that **HF** is *cumulative* hierarchy of finite sets.

3.1.4. Database queries

Database queries in general are just mappings of the types

$$q : \mathbf{HF}(\mathcal{A},\mathcal{U}) \to \mathbf{HF}(\mathcal{A},\mathcal{U}) \quad \text{or} \quad q : \mathbf{HF}(\mathcal{A},\mathcal{U}) \to \{\mathbf{true}, \mathbf{false}\}\ \text{[1]}, \tag{3.2}$$

the second case corresponding to *boolean queries*. Informally, $q(s) = a$ means for any $s, a \in \mathbf{HF}(\mathcal{A},\mathcal{U})$, that s is a *database state* and a is an *answer* to the *query* q about the state s, or a is an information extracted or (re)constructed from s by the query q. In the partial case of relational database, s is a set (or tuple) of relations and a is usually some new relation obtained some way from s. A boolean query just asks whether the given database state satisfies a certain property.

3.1.5. **Digression:** *Genericity of queries*

Let us assume for simplicity that $\mathcal{U} = \emptyset$ and that, moreover, $\mathcal{A}$ is a (possibly infinite) relational structure involving some relations including, say, relations 'A is substring of B' (to make search), 'A is of type B' (to deal with database schemas/types; e.g. '`Moscow` is of type `capital`', '`Moscow` is of type `word`', '`word` is of type `word`', '`22 May 2001` is of type `date`', '`2001` is of type `integer`', etc.) and, probably, a strict linear ordering $<_{\mathcal{A}}$ (cf. also Sect. 3.1.6). In any case, relations on $\mathcal{A}$, if the set of attributes is considered non-empty, should be included in our query language Δ; cf. Sect. 4. Actually, we will *not* officially include them, except of some notes like discussion of this section, because the main goal of this paper is different.

In the presence of $\mathcal{A}$ it makes sense to consider the requirement of *genericity* or *abstractness* of queries $q : \mathbf{HF}(\mathcal{A}) \to \mathbf{HF}(\mathcal{A})$ which consists of two clauses:

- q should *preserve support* of its arguments, i.e. if $a = q(\bar{s})$ then all attributes (labels) in $\mathcal{A}$ participating (at any depth) in a (the support of a) should also participate in $\bar{s}$ (no new elementary data arise);
- q *transforms isomorphic arguments to isomorphic results, respecting their supports.* That is

$$\bar{x} \cong \bar{y} \Rightarrow q(\bar{x}) \cong q(\bar{y}) \tag{3.3}$$

  with the latter isomorphism, if restricted to supports, *agreeing* with the former.

Here we consider isomorphism of $\mathbf{HF}(\mathcal{A})$-sets by taking into account all the history of their construction including participating of attributes in $\mathcal{A}$ and preserving of all relations given on $\mathcal{A}$.

[1] We can let $\{\mathbf{true}, \mathbf{false}\} \subseteq \mathbf{HF}$ by taking $\mathbf{false} = \emptyset$ and $\mathbf{true} = \{\emptyset\}$ or by considering $\{\mathbf{true}, \mathbf{false}\} \subseteq \mathcal{U}$.

More precisely, let $\simeq$ be arbitrary bijection (as a binary relation) between two subsets of $\mathcal{A}$ which additionally preserves all relations on $\mathcal{A}$ (such as $\underline{\text{is substring of}}$, $\underline{\text{is of type}}$, $<_{\mathcal{A}}$, etc.). That is, it is a *partial isomorphism* on $\mathcal{A}$. We write $A \simeq B$ if the attributes A and B are in this relation. Then this relation can be uniquely extended to a maximal partial bijection $\cong$ on $\mathbf{HF}(\mathcal{A})$ (depending on $\simeq$) by the following inductive clause:

- if $x = \{A_1 : x_1, \ldots, A_n : x_n\}$, $y = \{B_1 : y_1, \ldots, B_n : y_n\}$, $A_i \simeq B_i$ for all $i = 1, \ldots, n$ and it is already known that $x_i \cong y_i$ for all $i = 1, \ldots, n$, then let $x \cong y$.

In this case we say that sets x and y are *isomorphic* (w.r.t. the given $\simeq$ on $\mathcal{A}$). Evidently, $\emptyset \cong \emptyset$ according to this definition. Also the above requirement of 'respecting supports' for the generic queries proves to be superfluous under the given definition of isomorphism (if both occurrences of $\cong$ in (3.3) are, naturally, generated by the same $\simeq$).

The requirement of genericity for relational databases is well-known. For the case of **HF**-sets it was introduced in [45,46]. The case of **HFA** (do not mix with $\mathbf{HF}(\mathcal{A})$), i.e. of hyper-sets, is considered in [36] (rather similarly, but in terms of graphs and bisimulation of graphs, rather than of hyper-sets).

3.1.6. Pure **HF***-sets and the need of linear ordering*

For simplicity, the technical part of this paper is mostly devoted to pure **HF**-sets (without attributes and urelements: $\mathcal{A} = \mathcal{U} = \emptyset$).

For this case the notion of genericity from Sect. 3.1.5 degenerates, because isomorphism between pure **HF**-sets reduces to their equality.

Cf. also op. cit. to generalise the results of this paper to the case of attributes/labels, urelements and generic queries, i.e. to the case of presence of atomic data.

Note, that some natural *lexicographical* (canonical) linear ordering $<_{\mathbf{HF}}$ of **HF** (see formula (4.1) below) will play an essential role. Eventually, we need to imitate Turing machine tapes and to order computation steps when capturing PTIME or LOGSPACE computability over **HF** in a reasonable set-theoretic language (see the end of Sect. 4 and Sect. 5). Adapting considerations of this paper to the general case of $\mathbf{HF}(\mathcal{A},\mathcal{U})$ (as in [45,46]), some *linear orderings* $<_{\mathcal{A}}$ and $<_{\mathcal{U}}$ should be assumed in addition. Those might then be canonically extended to a linear ordering on the whole universe $\mathbf{HF}(\mathcal{A},\mathcal{U})$.

Here is a natural place where the reader could jump temporary to (the initial part of) Sect. 4. There we get down to an idea of a basic Δ-language over **HF** aimed to express queries of type (3.2) above (for pure **HF**). In any case, we will need to know what (PTIME or LOGSPACE) computable queries over **HF** are.

Therefore, we will fix some necessary details on encoding of **HF** sets in the rest of Sect. 3.

3.2. Computability of queries over **HF**

PTIME and, respectively, (N/D)LOGSPACE denote computability by a Turing machine in the time polynomial in the length of the input and, respectively, by a (nondeterministic/deterministic) Turing machine using the working tape of the length logarithmic in the length of the input. The typical inputs and outputs for a Turing machine are finite strings in a finite alphabet or, slightly more general (by using an appropriate evident encoding), finite graphs, etc.[2] It is well-known that DLOGSPACE $\subseteq$ NLOGSPACE $\subseteq$ PTIME, and it is an *open question* (with believably positive answer) whether $\subseteq$ are, in fact, proper inclusions [18].

There is a problem of some ambiguity of the notion of NLOGSPACE computability of functions in contrast to predicates: different nondeterministically chosen ways of computation may give different results. However, we will actually work in terms of an equivalent framework of FO$^{\otimes}_{\prec}$-definability described below in Sect. 3.5 where this ambiguity problem naturally disappears.

The key notion for this paper is the following definition of (say, PTIME or LOGSPACE) computability of operations or queries $q : \mathbf{HF}^m \to \mathbf{HF}$ over **HF**-sets, instead of finite strings of symbols.

Definition 3.2. Let ν : Codes $\to$ **HF** be any surjective encoding of **HF**-sets where Codes is any reasonable set of codes. (These will be some kind of finite graphs below.) We say that an operation (query) $q : \mathbf{HF}^m \to \mathbf{HF}$ is (PTIME, etc.) *computable with respect to* ν if the following diagram commutes for some (PTIME, etc.) computable transformation Q between codes:

$$\begin{tikzcd} \mathbf{HF}^m \arrow[r, "q"] & \mathbf{HF} \\ \mathrm{Codes}^m \arrow[u, "\nu^m"] \arrow[r, "Q"] & \mathrm{Codes} \arrow[u, "\nu"'] \end{tikzcd} \qquad q(\nu^m(\bar{c})) = \nu(Q(\bar{c})) \tag{3.4}$$

[2] A graph with vertices $v_1, \ldots, v_n$ may be represented on a Turing machine input tape (as usually in descriptive complexity theory) by 0-1-matrix $\{\epsilon_{ij}\}_{i,j=1}^{n}$ written row-by-row, where $\epsilon_{ij} = 1$ iff there exists an edge $v_i \to v_j$. Any finite first-order structure may be represented analogously. For definiteness and to avoid questions on what are abstract graphs, we could fix that vertices are always natural numbers $1, \ldots, n$.

for all $\bar{c} \in \text{Codes}$. Computability of boolean queries is defined analogously, via the diagram

$$\begin{array}{ccc} \mathbf{HF}^m & \xrightarrow{q} & \{\mathbf{true}, \mathbf{false}\} \\ {\scriptstyle \nu^m}\uparrow & & \uparrow{\scriptstyle id} \\ \text{Codes}^m & \xrightarrow{Q} & \{\mathbf{true}, \mathbf{false}\} \end{array} \qquad q(\nu^m(\bar{c})) = Q(\bar{c}) \; .$$

Using the set of Codes is essential here because Turing machine or any real computer does not "understand" immediately abstract **HF**-sets.

Computers deal with Codes. Peoples deal with abstract "high level" entities, say, with **HF**-sets. For example, this was, evidently, the idea behind introducing relational databases.

For the case of PTIME or (N/D)LOGSPACE we must denote corresponding classes of computable set-theoretic operations as $\mathcal{PTIME}_\nu$, $\mathcal{NLOGSPACE}_\nu$ or $\mathcal{DLOGSPACE}_\nu$ to emphasise that the corresponding notion of computability over **HF** *depends* on an encoding ν. Note, that for the case of *graph encoding* γ considered in Sect. 3.3 $\mathcal{NLOGSPACE}_\gamma$ or $\mathcal{DLOGSPACE}_\gamma$, unlike $\mathcal{PTIME}_\gamma$, are more *sensitive* to slight variations of this γ. At least, only for some specific versions of γ (which do not change the extent of $\mathcal{PTIME}_\gamma$) we can obtain reasonable notions and results for LOGSPACE computability over **HF**.

In general, let $\mathcal{C}$ denote some *notion of computability* over Codes, for example corresponding to some complexity class. More precisely, $\mathcal{C}$ is a recursive set of *programs* in a reasonable (probably resource restricted) programming language, such as the language of Turing machines (say, with some restrictions on working time or space as above). In particular, this means that, given a program $p \in \mathcal{C}$ and its input x, the result of applying p to x considered as a universal function $U(p, x) = p(x)$ is computable (not necessary in the sense of $\mathcal{C}$) in both arguments p and x. We associate with $\mathcal{C}$ the corresponding class of computable transformers $\text{Codes}^m \to \text{Codes}$, $m = 1, 2, \ldots$. This allows to define $\mathcal{C}_\nu$ as the class of $\mathcal{C}$-computable operations (and predicates) over **HF** with respect to any given $\nu : \text{Codes} \to \mathbf{HF}$.

Note, that for any given program $Q : \text{Codes}^m \to \text{Codes}$ (from a fixed class $\mathcal{C}$) it may be problematic to decide whether there exists (actually, a unique, if any) q making the diagram (3.4) commutative. I.e. Q may be "non-coherent" with ν.

Definition 3.3. Given an encoding $\nu : \text{Codes} \to \mathbf{HF}$ and a computability notion $\mathcal{C}$, we say that the class $\mathcal{C}_\nu$ of computable operations q over **HF** has an *effective syntax* relative to ν and is *captured* by this syntax if there exists a recursive (not necessary $\mathcal{C}$-computable) family of programs $Q_\pi \in \mathcal{C}$, $\pi = 0, 1, \ldots$, with all corresponding $q_\pi \in \mathcal{C}_\nu$, satisfying with Q_π the diagram (3.4), existing and exhausting the class $\mathcal{C}_\nu$. (Here the programs Q_π are not required to exhaust $\mathcal{C}$.)

Alternatively and more "realistically", we may let π range over formal expressions of a *language* L, instead of natural numbers. (We assume that L itself is a recursive set of expressions.) In this case q_π and Q_π may be considered, respectively, as *denotational* (over **HF**) and *operational* (over Codes) *semantics* of any expression (program) π in this language. In contrast to $\mathcal{C}$, each L-program π will have a corresponding q. In this paper we shall take in the role of L some appropriate versions Δ' of a natural set-theoretic language Δ (cf. Sect. 4) with a clear, self-evident denotational semantics and with *tractable* operational semantics over Codes (= some graphs), say, in terms of LOGSPACE computability.

> Thus, our goal is to find some reasonable encodings ν_1 and ν_2 and set-theoretic languages Δ_1 and Δ_2, with effective syntax relative to ν_1 and ν_2, and natural and coherent (as in (3.4)) denotational and operational semantics which capture exactly the classes $\mathcal{NLOGSPACE}_{\nu_1}$ and $\mathcal{DLOGSPACE}_{\nu_2}$, respectively.

As we argued above, we cannot avoid choosing some encodings ν_i here. Moreover, in principle there may be different choices for pairs $\langle \nu_i, \Delta_i \rangle$ corresponding to $\mathcal{NLOGSPACE}_{\nu_1}$ and $\mathcal{DLOGSPACE}_{\nu_2}$. This paper presents only one such reasonable choice.

3.3. Graph encoding of **HF**

An important example of Codes for **HF** is the class of all finite *acyclic pointed (rooted) graphs*[3] ($\mathcal{AG}$), i.e. graphs G with no cycles and with a *distinguished point (vertex)* p in each. (Note, that for the case of general universe $\mathbf{HF}(\mathcal{A},\mathcal{U})$ with attributes and urelements we should consider graphs with $\mathcal{A}$-labelled edges — more precisely, *labelled transition systems* where edges are formally triples $\langle u, A, v \rangle$ consisting of vertices u, v and a label A — and with some $\mathcal{U}$-labelled vertices having no children.) Then any computability notion $\mathcal{C}$ will determine a *class of computable graph transformers* $\mathcal{AG} \to \mathcal{AG}$.

Digression: To avoid questions what are abstract (or generic) computable graph transformers, we may adopt agreement of Footnote 2. What we really need in this

[3] Another interesting and essentially different version of Codes is the class of *finite trees* or, equivalently, of well-formed *bracket expressions* like $\{\,\}$, $\{\{\,\}\{\{\,\}\}\}$ etc. (Note some analogy of bracket expressions with XML [3].) This leads to another possible approach to PTIME, considered in [47], and to, yet non-considered, approach to LOGSPACE computability over **HF**. See also [49] where a general class of *polynomial regular* encodings ν was defined appropriate to considering corresponding $\mathcal{PTIME}_\nu$ over **HF**. This notion makes sense to extend to the case of *logarithmic space regular* encodings and to the case of hyper-sets (cf. Footnote 5).

paper of these transformers is just satisfying the diagram (3.4). Moreover, it is important that we impose no initial restriction on them. Otherwise how could we ever assert (without additional considerations) that we captured *full* (N/D)LOGSPACE or PTIME over **HF**?

However, this note does not mean that genericity of graph transformers is absolutely not related to our approach. In fact, graph transformers $Q : (\mathcal{EAG}_<^{(*)})^n \to \mathcal{EAG}_<^{(*)}$ (the class of graphs $\mathcal{EAG}_<^{(*)}$ is defined below) really used in the proof of the main Theorem 5.4 (on capturing (N/D)LOGSPCE over **HF**) to compute queries definable in corresponding version of query language Δ are *generic* in the sense that Q will preserve isomorphism of graphs: $\bar{G} \cong \bar{H} \Rightarrow Q(\bar{G}) \cong Q(\bar{H})$. Here the reader should take into account that these transformers are of a special kind: the vertices of the graph $Q(G_1, \ldots, G_n)$ are elements of some direct product $|G_1|^{k_1} \times \ldots \times |G_n|^{k_n}$ (cf. Sect. 3.5). Therefore there is some additional possibility to say that Q preserves isomorphisms *respecting the original one* $\bar{G} \cong \bar{H}$ (as for generic relational queries).

These generic or abstract in any sense graph transformers may be important from the point of view of implementation of our query language or as an intermediate theoretical concept needed for the implementation. Besides our logical description of abstract (N/D)LOGSPACE computable graph transformers in Sect. 3.5 it could be considered various other approaches to programming languages over graphs, e.g. Relational Machines [5] or Abstract State Machines [23,24,11,19][4]. In an unpublished paper (of a preliminary character) we already considered some abstract Graph Languages GL and GLB [37] especially created to a version of the query language Δ (cf. Sect. 4) and devoted rather to the case of Web-like databases and PTIME over **HF**.

Definition 3.4. Let $\gamma : \mathcal{AG} \to \mathbf{HF}$ be Mostowski's *general collapsing* operation (an encoding of **HF**-sets by graphs) which assigns a set $\gamma(G,p) \in \mathbf{HF}$ to each $\mathcal{AG}$ $\langle G, p \rangle$ in such a way that

$$\gamma(G,p) = \{\gamma(G,p') : p \to_G p' \text{ for some child } p' \text{ of } p \text{ in } G\} .$$

In particular, if p has no children in G then $\gamma(G,p) = \emptyset$. For a bit more complex example of G consisting just of three edges $p_3 \to p_2 \to p_1$ and $p_3 \to p_1$ we have $\gamma(G,p_1) = \emptyset$, $\gamma(G,p_2) = \{\emptyset\}$ and $\gamma(G,p_3) = \{\emptyset, \{\emptyset\}\}$. Note, that γ is *well defined* (in **HF**) because graphs $G \in \mathcal{AG}$ have no cycles.

[4] It is said in [11] that "codings of the hereditarily-finite sets by natural numbers play a central role" in [49]. First, let us note that we are using rather coding by graphs (via collapsing considered below or decoration when graphs are arbitrary and the universe consists of hypersets). Say, the traditional arithmetical Ackermann's encoding (4.3) is absolutely inappropriate from the point of view of PTIME or LOGSACE. On the other hand, it seems that direct using **HF**-sets without their encoding in an *abstract imperative computations* capturing so called *Choice-less PTIME* as in [11] does not release us from the necessity to represent them *someway* in a computer when implementing the approach of the op. cit. Finally, the approach of this paper, as well as of [49], gives an example of an *abstract, declarative and purely set-theoretic* description of PTIME and (N/D)LOGSPACE.

3.3.1. Agreement

We shall also write $p \in_G q$ instead of $q \to_G p$ and define formally any graph G as a *first-order relational structure* $\langle |G|, \in_G \rangle$ with $|G|$ as its set of vertices and the binary relation $\in_G$ for its edges. Then the above formula for γ can be rewritten more naturally

$$\gamma(G,p) = \{\gamma(G,p') : p' \in_G p\} \ . \qquad (3.5)$$

Sometimes we will apply γ to graphs with several kinds of edges (say, of various "colours"), i.e. with additional binary relations (such as a linear order $\prec_G$ on $|G|$), where $\in_G$ is just the *main* graph relation.

Even *more general*, we may consider that $\gamma(G,p)$ is defined also for *any* graph, not necessary acyclic. Just apply γ to the *initial acyclic (or well-founded)* part of G, denoted as $\mathbf{WF}(G)$. Here $\mathbf{WF}(G) \rightleftharpoons \langle W, \in_G\restriction_W \rangle$ with $W \subseteq |G|$ the least set of vertices such that for any given vertex $y \in |G|$ if all its children x, $y \to_G x$, are in W then we must have also $y \in W$.[5]

Let $\mathcal{G}$ be any class of finite pointed graphs, quite arbitrary or a special one such as $\mathcal{EG}$, $\mathcal{AG}$, $\mathcal{EAG}^*$, $\mathcal{EAG}^*_<$ or $\mathcal{EAG}^*_\prec$, etc. defined below. Then the restriction of γ to $\mathcal{G}$ defines a corresponding encoding of $\mathbf{HF}$-sets with Codes $= \mathcal{G}$ which will be called a *graph encoding* defined by $\mathcal{G}$. Thus, we may specify only $\mathcal{G}$ explicitly and assume the corresponding restricted γ implicitly.

Define *extensional* finite graphs ($\mathcal{EG}$) as those for which the ordinary set-theoretic *extensionality axiom* holds:

$$G \models \forall v \in x (v \in y) \ \& \ \forall v \in y (v \in x) \Rightarrow x = y \ ,$$

or, equivalently, different vertices in G must have different sets of children.

We will use abbreviations like $\mathcal{EAG}^*_\prec$, $\mathcal{EG}^*$, etc. with superscript $*$ and/or subscript $\prec$ to designate that the graphs considered involve additional relations for the (irreflexive) *transitive closure* $\in^*_G$ of the main graph relation $\in_G$

$$u \in^*_G v \rightleftharpoons u = u_0 \in_G u_1 \in_G \ldots \in_G u_n = v \quad \text{for some} \quad u_0, u_1, \ldots, u_n, \ n > 0 \ ,$$

and, respectively, for *any* (strict) linear order $\prec_G$ on the vertices of a graph G. Let also the subscript $<$ in $\mathcal{EAG}_<$ denote the *canonical linear order* $<_G$ on vertices of any $\mathcal{EAG}$ G which is inherited from the linear order $<_{\mathbf{HF}}$ on $\mathbf{HF}$ (defined

[5] There exists more radical approach (not considered in this paper officially) based on arbitrary graphs allowing cycles or infinite chains which leads to so called *hyper-sets* satisfying the *anti-foundation axiom* which asserts existence and uniqueness of γ satisfying (3.5) for *arbitrary* G. See [7] for general hyper-set theory and [48,49] for corresponding versions of bounded hyper-set theory and its Δ-language corresponding to PTIME over hyper-sets [35,36]. In particular this approach allows considering "cyclic" sets like $\Omega = \{\Omega\}$. Such (hereditarily-finite) hyper-sets may be used to represent so called Web-like Data Bases (WDB) i.e. databases organised analogously to World-Wide Web with arbitrary hyper-links; see [36]. It is also interesting to extend the results of the present paper for LOGSPACE from well-founded sets to hyper-sets.

below in (4.1); at this moment we do not need to know the precise definition) via collapsing $\gamma : |G| \to \mathbf{HF}$:

$$u <_G v \rightleftharpoons \gamma(u) <_{\mathbf{HF}} \gamma(v) \quad \text{for all} \quad u, v \in |G|.$$

Note, that for different vertices p_1, p_2 in $\mathcal{EAG}$ G we should evidently have $\gamma(G, p_1) \neq \gamma(G, p_2)$. Actually, any finite $G \in \mathcal{EAG}$ is isomorphic to finite *initial (i.e. transitive) part* of **HF**. (A part $T \subseteq \mathbf{HF}$ is called transitive if $u \in v \in T \Rightarrow u \in T$ for all $u, v \in \mathbf{HF}$.)

It can be shown [46] that $=$ and $\in$ over **HF** are PTIME computable w.r.t. encoding $\gamma : \mathcal{AG} \to \mathbf{HF}$. I.e. '$\gamma(G_1, p_1) = \gamma(G_2, p_2)$' and '$\gamma(G_1, p_1) \in \gamma(G_2, p_2)$' are decidable in PTIME for arbitrary $\langle G_1, p_1\rangle, \langle G_2, p_2\rangle \in \mathcal{AG}$. Say, equality is trivially decidable if one of p_1 or p_2 has no children. Then, assuming that for the children of p_1 and p_2 we can decide this equality, we can also easily decide it for p_1 and p_2, etc. All of this can be done in PTIME. But it is hardly possible to do in (N/D)LOGSPACE, unless (N/D)LOGSPACE = PTIME (due to PTIME completeness of the corresponding problem for $=$; cf. [14]).

It follows from Proposition 5.1 below that "restricting" an encoding $\gamma : \mathcal{AG} \to \mathbf{HF}$ to the class $\mathcal{EAG}_<$ (which is, strictly speaking, not a part of $\mathcal{AG}$), i.e. considering its corresponding version $\gamma : \mathcal{EAG}_< \to \mathbf{HF}$, makes this problem (of deciding $=$ and $\in$) easier — computable in NLOGSPACE. It becomes even DLOGSPACE computable in the case of $\mathcal{EAG}^*_<$, i.e. when the transitive closure $\in^*_G$ of the main graph relation is also immediately available. The additional relations $<_G$ and $\in^*_G$ on G help very much by giving more information on the input, the information which otherwise should be recovered by the cost of some computational resources.

Thus, we just change a "little" the initial notion of Codes $= \mathcal{AG}$ to the case of Codes $= \mathcal{EAG}_<$ or $= \mathcal{EAG}^*_<$, but this change proves to be crucial. This means that a database state should be represented (and permanently supported) not only by a graph G in $\mathcal{EAG}$, but also by its supplementary transitive closure and the canonical linear ordering. This additional information is the price for the efficiency (LOGSPACE instead of PTIME) of querying to such a database.

Digression *on complexity considerations.* Of course, from the practical point of view in the case of large databases this price is sufficiently high. We need a lot of additional space for $\mathcal{EAG}_<$ or $\mathcal{EAG}^*_<$ instead of simply $\mathcal{EAG}$. But, otherwise, we would rather have to do with PTIME instead of (N/D)LOGSPACE so that, anyway, PTIME-compitability would require, in general, polynomial space. In the case of $\mathcal{EAG}_<$ or $\mathcal{EAG}^*_<$ we just "pass" this space to the input databse state at the very beginning. It is a real practical dilemma, what is better. In the real computer terms, the price is the size of *hard disk(s)* needed for the current state of database which seems to be not so high for the contemporary powerful computers. It seems more serious the price of how often we will be forced to address to the hard disk for this additional auxiliary information. On the other hand, logarithmic space (which is practically bounded by (hopefully) not so big constant) gives a hope that the real computer "quick" *operative memory* will be enough. This can be a serious adventage. It would be interesting to undertake more careful analysis. However, our goal here is rather theoretical one, based on the usual complexity theory notions. We should also realize that the practice may put forward quite unexpected problems weakly related with our complexity theoretic considerations. Say, in the case of Web-like databases [50] each step of browsing costs so high that any general considerations on PTIME or LOGSPACE may have no value as real complexity measures. The role of capturing PTIME or LOGSPACE by some query languages we are concerned here is rather that of characterising their *expessive power*, not of the (real life) complexity.

It is evidently inevitable considering something like the above restricted to $\mathcal{EAG}$ and further enriched to $\mathcal{EAG}^*_<$ classes of graphs. We could avoid this complication if (N/D)LOGSPACE = PTIME were true, or if $<_G$ and $\in^*_G$ were formally *definable* by suitable means sufficient for our goals (say, in the language FO$^\odot$ considered in Sect. 3.5; we have only a partial, definability result of Proposition 5.1).

Actually, this paper is devoted to machine independent descriptions of two classes of computable set theoretic operations: $\mathcal{NLOGSPACE}_{\mathcal{EAG}_<}$ and $\mathcal{DLOGSPACE}_{\mathcal{EAG}^*_<}$ for the graph encoding γ restricted to corresponding classes of graphs $\mathcal{EAG}_<$ and $\mathcal{EAG}^*_<$.

Compare this note with the corresponding framed one on the page 10. Cf. also Figures 5.1–5.4 below demonstrating LOGSPACE-computable graph transformers corresponding, relative to γ, to some set-theoretic operations.

3.4. Further agreement on relational presentation of graphs

It makes sense to further represent graphs in $\mathcal{EAG}^{(*)}_<$ by those in $\mathcal{AG}^{(*)}_{=,<}$ [6] and to consider any such graph G in the form of a finite relational structure

$$G = \langle |G|; \in_G, \in^*_G, =_G, <_G, V_G \rangle$$

with $\in^*_G$ and $<_G$ probably omitted depending on the considered class of graphs such as $\mathcal{EAG}_<$ or $\mathcal{EAG}^*$. Here $=_G$ is one more predicate on the finite set $|G|$ which is required to be an *equivalence relation* and V_G is a unary predicate on $|G|$ representing just (part of) one equivalence class under $=_G$ and containing the distinguished vertex of the graph. We need V_G as an unary relation rather than as an individual constant because we use purely relational language and, if

[6] Here and below we use $(*)$ in parantheses to denote two cases: with and without $*$.

necessary, can define V_G by a logical formula. Moreover, *we always assume that for G in $\mathcal{AG}^*_{=,<}$*

$$\left.\begin{array}{l} u =_G v \rightleftharpoons \gamma(G,u) =_{\mathbf{HF}} \gamma(G,v)\ , \\ u <_G v \rightleftharpoons \gamma(G,u) <_{\mathbf{HF}} \gamma(G,v)\ , \\ \in^*_G \text{ is the transitive closure of } \in_G \text{ and} \\ V_G \neq \emptyset \text{ is part of an equivalence class of } =_G\ . \end{array}\right\} \tag{3.6}$$

Evidently, $=_G$ is a congruence w.r.t. $<_G$ [7]. The relations $\in_G$, $\in^*_G$ and V_G can be easily "improved" to

$$\begin{array}{l} u \mathrel{\hat\in_G} v \rightleftharpoons \exists u' \in_G v(u =_G u')\ , \\ u \mathrel{\hat\in^*_G} v \rightleftharpoons \exists u' \in^*_G v(u =_G u')\ , \\ u \in \hat V_G \rightleftharpoons \exists u' \in V_G(u =_G u') \end{array}$$

so that $=_G$ will become a *full* congruence because

$$\begin{array}{l} u \mathrel{\hat\in_G} v \iff \gamma(G,u) \in_{\mathbf{HF}} \gamma(G,v)\ , \\ u \mathrel{\hat\in^*_G} v \iff \gamma(G,u) \in^*_{\mathbf{HF}} \gamma(G,v)\ . \end{array}$$

These equivalences and also the fact that $\hat\in^*_G$ is really the transitive closure of $\hat\in_G$ and that G under $\hat\in_G$ is still acyclic graph follow from the equivalence

$$x \in^n_{\mathbf{HF}} \gamma(G,v) \iff x = \gamma(G,w) \text{ for some } w \in^n_G v$$

which can be easily proved by induction on $n \geq 1$. Therefore we can always consider corresponding quotient $\hat G \rightleftharpoons G/_{=_G} \in \mathcal{EAG}^{(*)}_<$ for any $G \in \mathcal{AG}^{(*)}_{=,<}$ (satisfying, by definition, the assumptions (3.6)).

If $=_G$ is a full congruence in G (not only w.r.t. $<_G$) then the above "improving" gives nothing:

$$\begin{array}{l} u \in_G v \iff \exists u' \in_G v(u =_G u')\ , \\ u \in^*_G v \iff \exists u' \in^*_G v(u =_G u')\ , \end{array}$$

and, moreover, G is extensional *up to* $=_G$:

$$u =_G v \iff \forall u' \in_G u \exists v' \in_G v(u' =_G v') \ \&\ \forall v' \in_G v \exists u' \in_G u(u' =_G v')\ .$$

Anyway, it follows that graphs in $\mathcal{EAG}^{(*)}_<$ may be represented by graphs in $\mathcal{AG}^{(*)}_{=,<}$ as their quotients. Moreover, as graphs are intended to represent **HF**-sets, it is important that for each equivalence class $[v]$ of a vertex v we also have

$$\gamma([v]) = \gamma(v)$$

what can be easily shown by $\in$-induction over acyclic graphs. Of course, for the quotient graph $\hat G \in \mathcal{EAG}^{(*)}_<$ we also have

[7] i.e. it is such equivalence relation $=_G$ that $G \models (x = x' \ \&\ y = y' \ \&\ x < y \Rightarrow x' < y')$

- $[u] \in_{\hat{G}} [v] \iff u \in_G v$ for some/any representatives u and v,
- $\in^*_{\hat{G}}$ is the transitive closure of $\in_{\hat{G}}$ and
- $<_{\hat{G}}$ is the canonical linear ordering on $\hat{G}$.

Assuming that $=_G$ are (full) congruence relations, we should consider an isomorphism between G_1 and G_2 rather to be a relation $\leftrightarrow \subseteq |G_1| \times |G_2|$ which preserves all relations on these graphs, including $=_{G_1}$ and $=_{G_2}$, and the distinguished elements (classes) V_{G_1} and V_{G_2}. This is actually an isomorphism *up to* the congruence relations $=_{G_i}$, in comparison with the ordinary (bijective) isomorphisms. Evidently, for ($\in$-)isomorphic graphs $\langle G_1, p_1\rangle$ and $\langle G_2, p_2\rangle$ we have $\gamma(G_1, p_1) = \gamma(G_2, p_2)$.

3.5. Definability of graph transformers by logical means

Instead of considering (LOGSPACE or other) *computability* of graph transformers $Q : \bar{G} = \langle G_1, \ldots, G_n\rangle \longmapsto G$ (determined up to a graph isomorphism) needed in the diagram (3.4) we will deal with their *definability* in a formal language. To this end we may use, say, some appropriate extension FO$'$ of the first-order language

$$\mathrm{FO}(=_1, \ldots, =_n; \in_1, \ldots, \in_n; \in^*_1, \ldots, \in^*_n; <_1, \ldots, <_n; V_1, \ldots, V_n)$$

over a tuple of graphs $\langle G_1, \ldots, G_n\rangle$ from $\mathcal{AG}^{(*)}_{=,<}$ considered as many-sorted structure (with $=_i$, $\in_i$, $\in^*_i$, $<_i$ and V_i interpreted, respectively, as $=_{G_i}$, $\in_{G_i}$, $\in^*_{G_i}$, $<_{G_i}$ and V_{G_i}; cf. the agreement of Sect. 3.4). It is reasonable to consider for the resulting graph

$$G = \langle |G|; =_G, \in_G, \in^*_G, <_G, V_G\rangle$$

that $V_G \subseteq |G| \subseteq |\tilde{G}| \rightleftharpoons |G_1|^{k_1} \times \cdots \times |G_n|^{k_n}$, $=_G \subseteq |\tilde{G}| \times |\tilde{G}|$, $\in^{(*)}_G \subseteq |\tilde{G}| \times |\tilde{G}|$ and $<_G \subseteq |\tilde{G}| \times |\tilde{G}|$, for some $k_1, \ldots, k_n \geq 1$, and define $|G|$, V_G, $=_G$, $\in^{(*)}_G$ and $<_G$ by FO$'$-formulas

$$|G|(\tilde{v}), V_G(\tilde{v}) \text{ (or } \tilde{v} \in |G|, \tilde{v} \in V_G),\ \tilde{u} =_G \tilde{v}, \tilde{u} \in^{(*)}_G \tilde{v} \text{ and } \tilde{u} <_G \tilde{v}$$

of $k = k_1 + \ldots + k_n$ and $2k$ free variables, respectively. Actually, we may take $=_G$ to be a *partial equivalence relation* (i.e. a transitive and symmetric relation, even a congruence relation; cf. Sect. 3.4) on $|\tilde{G}|$ and define

$$|G|(\tilde{v}) \rightleftharpoons \tilde{v} =_G \tilde{v}\ .$$

This approach to formal definability of graph transformers relies on the well known concept of first-order interpretations between theories or structures (cf.

[26,51]).[8] Actually we will assume *for the input graphs* that $=_{G_i}$ *are just identities*.[9]

3.6. FO$^{\otimes}$ *and* FO$^{\odot}$

We will actually use an extension $\mathrm{FO}' = \mathrm{FO}^{\otimes}$ of the language FO by a new predicate forming construct $[\lambda\bar{x}\bar{y}.\varphi(\bar{x},\bar{y},\bar{z})]^{\otimes}$ for any definable (in FO$^{\otimes}$) relation $\lambda\bar{x}\bar{y}.\varphi(\bar{x},\bar{y},\bar{z})$ of two lists of variables $\bar{x}$ and $\bar{y}$ of the same length. Other free variables $\bar{z}$ of φ are considered as parameters. We call this construct the "horizontal" *transitive closure* and define its meaning in any G by

$$[\lambda\bar{x}\bar{y}.\varphi(\bar{x},\bar{y})]^{\otimes}(\bar{u},\bar{v}) \quad (\text{or } \varphi^{\otimes}(\bar{u},\bar{v})) \quad \text{iff} \quad \varphi(\bar{u}_0,\bar{u}_1),\ldots,\varphi(\bar{u}_{n-1},\bar{u}_n) \quad \text{hold}$$

for some $n > 0$ and $\bar{u} = \bar{u}_0,\ldots,\bar{u}_n = \bar{v} \in |G|$. It may *arbitrarily* participate in any other FO$^{\otimes}$-formula as a predicate.

Note, that to be $\mathcal{EAG}$ is evidently an FO$^{\otimes}$-definable property of graphs. See also Remark 5.3 in Sect. 5 for the case of $\mathcal{EAG}^{(*)}_{<}$.

We may consider also a *deterministic* version $\odot$ of transitive closure $\otimes$ which works as $\otimes$ for any formula $\varphi(\bar{x},\bar{y})$ when it defines a "deterministic" (partial) *mapping* $\bar{x} \longmapsto \bar{y}$ [10]. Otherwise, applying $\odot$ gives, say, the false predicate (or, alternatively, $\odot$ preliminary corrects φ to make it "deterministic").

FO$^{\otimes}$ and FO$^{\odot}$ are essentially identical to the languages FO + TC and FO + DTC, respectively, considered in [26,27]. (Actually, in [26,27] TC and DTC coincide with the *reflexive* versions of these $\otimes$- and $\odot$-constructs.) However, the denotations TC and $*$ will be occupied in our paper for the "vertical" transitive closure in the universe of sets **HF**:

$$u \in \mathrm{TC}(v) \rightleftharpoons u \in^{*} v \rightleftharpoons [\lambda xy.x \in y]^{\otimes}(u,v) \ .$$

I.e. we use TC and $*$ just in connection with the relation $\in$. Evidently, $\in^{*}_{G}$ coincides with $\in^{\otimes}_{G}$.

[8] We will use tuples of variables $\tilde{u},\tilde{v},\tilde{w},\ldots$ over $|\tilde{G}|$ with *tilde* (instead of usual bared variables $\bar{u},\bar{v},\bar{w},\ldots$) only in the manner of this Section to denote vertices of the logically defined graph G. We will also use in analogous situations *hat*-tuples $\hat{u}$, etc. for tuples of variables in which the above tilde-tuples participate, say, if $\hat{u} \rightleftharpoons x,\tilde{y},\tilde{z},v,\bar{w}$.

[9] Note, that if initial $=_{G_i}$ are just identities then (after "improving" $\in_G, \in^{*}_G$ and V_G as in Sect. 3.4) by using linear orderings $<_{G_i}$ we can choose in each equivalence class of $=_G$ in the resulting graph G the lexicographically least representative tuple $\tilde{v}$ thereby making $=_G$ a (partial) *identity* instead of equivalence, etc. It follows that we could consider graphs more directly without assuming any quotients. But this additional step would somewhat complicate our construction (in the proof of Theorem 5.4). That is why we adopt more general agreement of Sect. 3.4.

[10] This means that $\varphi(\bar{x},\bar{y})$ & $\varphi(\bar{x},\bar{z}) \Rightarrow \bar{y} = \bar{z}$ holds, and satisfying $\forall\bar{x}\exists\bar{y}\varphi(\bar{x},\bar{y})$ is not required.

3.7. Recalling some results from descriptive complexity theory over finite models

It is proved in [26] that the notion of definability in $\mathrm{FO}_\prec$ + *positive*[11] $\otimes$ in finite structures *linear ordered* by $\prec$ is equivalent to NLOGSPACE computability. Moreover, $\mathrm{FO}_\prec$ + *positive* $\otimes$ has the same expressive power in these structures as the *full* $\mathrm{FO}_\prec + \otimes$, i.e. as $\mathrm{FO}^\otimes_\prec$ [27]. This result is equivalent to the statement Co-NLOGSPACE = NLOGSPACE (cf. also [52]) which has been widely believed to be false previously.

> That is $\mathrm{FO}^\otimes_\prec$, which is closed under negation, captures exactly NLOGSPACE. Analogously, $\mathrm{FO}^\odot_\prec$ captures DLOGSPACE, but here the closure under negation is not so problematic. (In particular, we have FO $\subseteq$ DLOGSPACE.) Therefore, we may freely interchange the notions $\mathrm{FO}^{\otimes/\odot}_\prec$ and (N/D)LOGSPACE. Moreover, we may also consider that the notion of N(D)LOGSPACE *computable graph transformers* $\mathcal{EAG}^{(*)}_< \to \mathcal{EAG}^{(*)}_<$ is defined in terms of $\mathrm{FO}^\otimes$ (respectively, $\mathrm{FO}^\odot$) due to Sect. 3.5 and also due to Remark 5.3 below which says that '$G \in \mathcal{EAG}^{(*)}_<$' is FO-definable.

We will need also the well-known extension FO + LFP of FO by the *least fixed point* construct

$$\textbf{the-least}\ P.\,[P(\bar{x}) \leftrightarrow \varphi(\bar{x}, P)] \tag{3.7}$$

with the predicate variable P occurring in φ positively (cf. Footnote 11). The semantics of this construct is based on an iterative computation of the least predicate P satisfying the condition in the square brackets in (3.7). (Start with $P_0 \rightleftharpoons \emptyset$ and continue by $P_{n+1} \rightleftharpoons \lambda\bar{x}.\varphi(\bar{x}, P_n)$, up to stabilisation $P \rightleftharpoons P_n = P_{n+1}$ of this monotonic sequence.) It actually subsumes $\otimes$ and $\odot$. It was shown in [25,38, 53] that definability in $\mathrm{FO}_\prec$+LFP over finite (linear ordered!) structures exactly corresponds to PTIME computability. Cf. also analogous results for PTIME [41,21] and DLOGSPACE [21] in terms of recursive or, respectively, primitive recursive (global) functions, instead of predicates, in finite segments of natural numbers.

4. Δ-languages of set theoretic operations

4.1. Basic Δ-language for the pure **HF** [12]

Definition 4.1. Define inductively Δ^*-*formulas* and Δ^*-*terms* by the clauses

$$\langle\Delta^*\text{-terms}\rangle ::= \langle\text{variables}\rangle \mid \emptyset \mid \{a, b\} \mid \bigcup a \mid \{t(x) : x \in^{(*)} a \;\&\; \varphi(x)\}$$

[11] i.e. occurring under even number of negations (taking into account that $\varphi \Rightarrow \psi$ is $\neg\varphi \vee \psi$)
[12] See [46] for generalisation of Δ for the case of $\mathbf{HF}(\mathcal{A}, \mathcal{U})$.

$$\langle\Delta^*\text{-formulas}\rangle ::= a{=}b \mid a{\in^{(*)}}b \mid \varphi \,\&\, \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \forall x{\in^{(*)}}a\varphi(x) \mid \exists x{\in^{(*)}}a\varphi(x)$$

where φ and ψ are any Δ^*-formulas, a, b and t are any Δ^*-terms and x is a variable not free in a. The parentheses around $*$ in $\in^{(*)}$ mean that there are two versions of the membership relation: $\in$ and its transitive closure $\in^*$. Then Δ_0^*-*formulas* are defined as those Δ^*-formulas involving only atomic terms (i.e. just variables or the constant $\emptyset$. We write Δ (Δ_0; cf. [33])) when $\in^*$ is not used at all. The sub-language Δ corresponds to the *basic* [17] or *rudimentary* [29] set-theoretic operations.

From the database's point of view, the language Δ is somewhat analogous to *relational calculus*, however it is devoted to sets rather than to relations.

Using the term Δ is not completely fixed in our different papers, but the main idea is that Δ-formulas involve only *bounded* quantification $\forall x \in^{(*)} a$ and $\exists x \in^{(*)} a$. That is why, according to traditions of mathematical logic and set theory, we use the name Δ for the language above and for various of its versions. All other constructs of Δ are also in a sense "bounded". But, for example, the unrestricted *power-set operation* $\mathcal{P}(x) \rightleftharpoons \{y : y \subseteq x\}$ is considered as intuitively "unbounded", "impredicative". Note, that $\mathcal{P}(x)$ is also evidently related with exponentiation, and therefore it is computationally intractable. (It would lead us to Kalmar's elementary computability rather than to PTIME or LOGSPACE.) In general, let Δ' denote some reasonable, still intuitively "bounded", extension of the class of basic operations.

For simplicity, as common, we shall use set variables, Δ^*-terms and formulas *both as syntactic objects and as denotations of their values* in **HF**. For example, Δ-separation $\{x \in a : \varphi(x)\}$ for $\varphi \in \Delta$ gives 'the set of all x in the set a for which $\varphi(x)$ holds' and is a special case of the construct $\{t(x) : x \in a \,\&\, \varphi(x)\}$ = 'the set of all values of $t(x)$ such that ...'. Also $x \in \{a, b\}$ iff $x = a$ or $x = b$, $x \in \bigcup a$ iff $\exists z \in a(x \in z)$ and $\in^*$ is a transitive closure of the membership relation $\in$ on **HF**, i.e. $x \in^* y$ iff $x \in x_1 \in x_2 \in \ldots \in x_n \in y$ for some $n \geq 0$ and $x_1, \ldots, x_n$ in **HF**. In particular, define in Δ^* the *transitive closure of a set* y as $\mathrm{TC}(y) \rightleftharpoons \{x : x \in^* y\}$. The meaning of logical symbols & ('and'), $\vee$ ('or'), $\neg$ ('not'), $\forall$ ('for all'), $\exists$ ('exists') is well known. Bounded quantifiers $\forall x \in^{(*)} a$ and $\exists x \in^{(*)} a$ have the same meaning as the ordinary unbounded ones except the variable x ranges only over the (transitive closure of the) set (denoted by) a.

Note that bounded quantifiers may be eliminated from $\Delta^{(*)}$-language:

$$\begin{aligned} \exists x \in^{(*)} a\psi(x) &\iff \{x \in^{(*)} a : \psi(x)\} \neq \emptyset \,, \\ \forall x \in^{(*)} a\psi(x) &\iff \{x \in^{(*)} a : \neg\psi(x)\} = \emptyset \,. \end{aligned}$$

Any Δ'-term $t(\bar{x})$ evidently defines a set-theoretic operation

$$\lambda\bar{x}.t(\bar{x}) : \mathbf{HF}^n \to \mathbf{HF}\ .$$

For example, let $u \cup v \rightleftharpoons \bigcup\{u, v\}$, $u \cap v \rightleftharpoons \{x \in u : x \in v\}$, $u \setminus v \rightleftharpoons \{x \in u : x \notin v\}$, $\{u\} \rightleftharpoons \{u, u\}$, $\{u_1, u_2, \ldots, u_k\} \rightleftharpoons \{u_1\} \cup \{u_2\} \cup \ldots \cup \{u_k\}$. Ordered singletons, pairs, triples, as well as the predicate "x is an ordered pair" etc. are defined in Δ as $\langle u\rangle \rightleftharpoons u$, $\langle u, v\rangle \rightleftharpoons \{\{u\}, \{u, v\}\}$, $\langle u, v, w\rangle \rightleftharpoons \langle\langle u, v\rangle, w\rangle$, $\langle \bar{y}, z\rangle \rightleftharpoons \langle\langle\bar{y}\rangle, z\rangle$, etc. It follows from the definitions that $\bigcup\langle u, v\rangle = \{u, v\}$ and for any set of ordered pairs r the set of all the components of these pairs can be defined in Δ as $\mathbf{field}(r) \rightleftharpoons \bigcup\bigcup r$ (union of union of the set r).[13]

As in [46], we may define in the Δ-language many other useful operations on sets, e.g. the *Cartesian product* $A \times B$, *Cartesian power* A^k and *disjoint unions* $A + B$ and $\sum_{i \in I} A_i$ of any two sets A and B and of any Δ-definable family of sets A_i, etc.[14]

As usual, any set g of ordered pairs may be considered as a (directed) graph. Moreover, for arbitrary set g we may just ignore its elements which are not ordered pairs and therefore consider it as a graph. Any pair $\langle g, p\rangle \in \mathbf{HF}$ with a graph g and its vertex p is just a *pointed graph* in the framework of set theory. If $p \notin \mathbf{field}(g)$ then it is considered as an *isolated* vertex.

We will use extensively the fact that any $\Delta^{(*)}$-formula φ is equivalent to a $\Delta_0^{(*)}$-formula [17]. Cf. also [44,49] for corresponding proof-theoretic considerations and formal *reductions* (as in lambda calculus) with $\Delta_0^{(*)}$-formulas being *normal forms* for $\Delta^{(*)}$-formulas. For example, the formula $\langle u, v\rangle = w$ is equivalent to Δ_0-formula (the normal form)

$$\begin{aligned}\exists s, p \in w(\forall x \in w(x = s \vee x = p)\ \&\ u \in s\ \&\ u, v \in p\ \&\\ \forall x \in s(x = u)\ \&\ \forall x \in p(x = u \vee x = v))\end{aligned}$$

and may serve as its abbreviation. Then $\langle u, v\rangle \in z$ is equivalent to $\exists w \in z(\langle u, v\rangle = w)$ and therefore also reducible to Δ_0. Analogously, we may find expressions for $w = \bigcup u$, etc.[15]

[13] Here the double union $\bigcup\bigcup$ is related with the above specific definition of ordered pairs. Note, that in the case of $\mathbf{HF}(\mathcal{A})$-sets with attributes $\mathcal{A}$ the ordered pair $\langle u, v\rangle$ could be defined more naturally from a practical point of view as $\{\mathrm{Fst} : u, \mathrm{Snd} : v\}$ for some two special labels (attributes) Fst and Snd.

[14] It can be shown [43,44,49] that Δ defines exactly all *provably-recursive set-theoretic operations* in Kripke-Platek set theory [8] without foundation axiom (and analogously for extensions of Δ). In this sense this language is sufficiently complete from the point of view of its expressive power.

[15] Note that Δ_0^*-formulas are just first-order formulas with bounded quantifiers in the language $\in, \in^*$. By this reason and by the above considerations Δ^*, involving rather complex set-theoretic operations, may be also considered as an *analogue of first-order logic language*. There is also corresponding proof-theoretic conservativeness result [44,49], relative to Δ_0-formulas, for Kripke-Platek set theory without foundation axiom (exactly corresponding to Δ in the

Remark 4.2. Such abbreviations for $\langle u,v\rangle = w$, $w = \bigcup u$, $u = \emptyset$, $u = \{\emptyset\}$, etc. may be also used in the framework of the language $\mathrm{FO}(\in,=)$ (instead of Δ_0) when working in a graph G in $\mathcal{EAG}$ (instead of **HF**).

4.2. Extensions of Δ^*

However, there are notions which are hardly definable in Δ^*, such as the *canonical* or *lexicographical linear ordering* $<_{\mathbf{HF}}$ on **HF** uniquelly characterised by the

Axiom for $<_{\mathbf{HF}}$**:**

$$x < y \iff \exists v \in y \setminus x(\{w \in x : v < w\} = \{w \in y : v < w\})\ . \tag{4.1}$$

Actually the same formula defines canonical linear ordering $<_G$ on vertices of any G in $\mathcal{EAG}$ so that $u <_G v \iff \gamma(G,u) <_{\mathbf{HF}} \gamma(G,v)$. Note, that (4.1) defines $<_{\mathbf{HF}}$ and $<_G$ uniquely, but only *implicitly*. We do not know whether it is definable *explicitly* in the considered versions of Δ (except of $\Delta^* + \mathbf{Rec}$; cf. (4.4) below). We will need also another, equivalent version of this axiom:

$$\begin{aligned} x < y \iff & x \subset y \vee \exists u \in x \exists v \in y \\ & (\{w \in x : u < w\} = \{w \in y : v < w\}\ \&\ u < v) \end{aligned}$$

or even (slightly more complicated)

$$\begin{aligned} x < y \iff & x \subset y \vee \exists u,v(u \in x\ \&\ v \in y\ \&\ x \neq y\ \&\ x \not\subset y\ \&\ y \not\subset x \\ & \{w \in x : u < w\} = \{w \in y : v < w\}\ \&\ u \neq v\ \&\ u < v)\ . \end{aligned} \tag{4.2}$$

The extra conjuncts, evidently, do not matter here. However, they will play some role later.[16]

This ordering on **HF** actually has the type of natural numbers and can be easily shown to coincide with that induced by Ackermann's bijection $\mathbf{Ack} : \mathbf{N} \to \mathbf{HF}$, $\mathbf{N} = \{0,1,2,\ldots\}$ — the most popular encoding of the universe **HF** by natural numbers:

$$\mathbf{Ack}(2^{n_1} + 2^{n_2} + \ldots + 2^{n_j}) = \{\mathbf{Ack}(n_1), \mathbf{Ack}(n_2), \ldots, \mathbf{Ack}(n_j)\} \tag{4.3}$$

for any $j \geq 0$ and $n_1 > n_2 > \ldots > n_j \geq 0$. In particular, $\mathbf{Ack}(0) = \emptyset$. Note that the very simple singleton operation $x \longmapsto \{x\}$ corresponds to the arithmetical exponential $k \longmapsto 2^k$ with respect to **Ack**, which is surely intractable. It follows that we must consider, instead of **Ack**, other encodings for the universe **HF** like the graph encoding (collapsing) γ described in the Sect. 3.3.

sense of Footnote 14) over set theory consisting only of Extensionality Axiom (which is also "almost" first-order logic).

[16] Cf. formulas (5.1), (5.2) and corresponding comments in the proof of Proposition 5.1.

In contrast to the general recursion theory, here, in the context of complexity theory and of possible applications to databases, we should be much more careful in choosing an appropriate encoding (representation in a computer) of **HF**-sets.

We may extend Δ also by a set-theoretic *collapsing* operation

$$\mathrm{C} : \mathbf{HF} \to \mathbf{HF}$$

which is a "restriction" of the general collapsing γ to the case of (pointed) $\mathcal{EAG}$s considered as elements of the universe **HF**. Thus, for any $\langle g, p\rangle \in \mathbf{HF}$, $\mathrm{C}(\langle g, p\rangle)$ is the set in **HF** corresponding to the point p of g under γ. If g is not an extensional or not an acyclic graph, then let $\mathrm{C}(\langle g, p\rangle) = \emptyset$.

The collapsing operation plays a fundamental role. It reflects in the framework of set theory our informal "understanding" of sets of sets, etc. in terms of graphs.

From the database point of view $\langle g, p\rangle$ is a data representing a "plan" of constructing another data, i.e. a set corresponding to the vertex p of g. Collapsing is just a *legal possibility to fulfil such plans* which a reasonable query language should give.

This is related with *two levels of representing data* in a set-theoretic database: (i) comparatively low level — via graphs (in general case, with labelled edges), and (ii) high, abstract level — via **HF**-sets. The operation of collapsing relates both these levels in terms of **HF**.

We will need two other *weaker* (because they need *more* from their arguments) versions of C:

- $\mathrm{C}^{\odot}(\langle g, <_g, \in_g^*, p\rangle) \rightleftharpoons \mathrm{C}(\langle g, p\rangle)$ if g is $\mathcal{EAG}$, $<_g$ is the canonical linear order on the vertices of g and $\in_g^*$ is the transitive closure of $\in_g = \{\langle u, v\rangle : \langle u, v\rangle \in g\}$; $\mathrm{C}^{\odot}(\langle g, <_g, \in_g^*, p\rangle) \rightleftharpoons \emptyset$, otherwise. I.e. $\mathrm{C}^{\odot}$, unlike C, is properly defined only on $\mathcal{EAG}^*_< \subseteq \mathbf{HF}$, i.e. on the canonically ordered $\mathcal{EAG}^*$s.
- $\mathrm{C}^{\otimes}(\langle g, <_g, p\rangle) \rightleftharpoons \mathrm{C}(\langle g, p\rangle)$ if g is $\mathcal{EAG}$ and $<_g$ is the canonical linear order on the vertices of g; $\mathrm{C}^{\otimes}(\langle g, <_g, p\rangle) \rightleftharpoons \emptyset$, otherwise. I.e. $\mathrm{C}^{\otimes}$, unlike C and $\mathrm{C}^{\odot}$, is properly defined only on $\mathcal{EAG}_< \subseteq \mathbf{HF}$, i.e. on the canonically ordered $\mathcal{EAG}$s.

The reason of using the superscripts $\otimes$ and $\odot$ in C will become obvious later in Theorem 5.4. (Note that $\mathrm{C}^{\otimes}$ was called $\hat{\mathrm{C}}$ in [34].)

Another particularly important construct is *recursive bounded separation* **Rec**:

$$\textbf{the-least}\ p.\,[p = \{x \in a : \varphi(x, p)\}]$$

which could be additionally included in Definition 4.1 if we would interested here in PTIME rather than LOGSPACE. It must be considered as a *term* which denotes the least solution $p \subseteq a$ of the equation in square brackets for φ a bounded formula (satisfying a reasonable "positivity" condition for the set variable p).

We will also consider an extension of our Δ^*-language by new kind of terms $r^\otimes$ (instead of formulas, as for the case of FO in Sect. 3.5) denoting the "horizontal" *transitive closure* of any relation (set of pairs) r. (Do not mix denotations $r^\otimes$ and $r^\odot$ with $\mathrm{C}^\otimes$ and $\mathrm{C}^\odot$ defined above and having a completely different meaning.) It will be called $\Delta^\otimes$, with corresponding $\Delta^\otimes$*-terms* and $\Delta^\otimes$*-formulas*. Evidently, $\otimes$ is definable by **Rec**.[17] The *deterministic* version $\odot$ of $\otimes$ is formally defined as

$$r^\odot \rightleftharpoons \mathrm{Det}(r)^\otimes \text{ where } \mathrm{Det}(r) \rightleftharpoons \{\langle x,y\rangle \in r : \exists! z(\langle x,z\rangle \in r)\} .$$

Analogously, we use denotations like $\Delta_<$ or $\Delta_<^\otimes$ if the predicate for $<_{\mathbf{HF}}$ is included in the language.

The following characterisation of the language $\Delta^* + \mathbf{Rec} + \mathrm{C}$ has been obtained in [42,44,46].

Theorem 4.3. *The class of operations over* **HF** *definable in the language* $\Delta^* + \mathbf{Rec} + \mathrm{C}$ *coincides with the class* $\mathcal{PTIME}_\gamma$ *of operations computable in polynomial time under graph representation* $\gamma : \mathcal{AG} \to \mathbf{HF}$ *of* **HF**-*sets or, equivalently, computable by graph transformers definable in* FO + LFP. □

It is essentially used in the proof of this theorem that [46]

$$<_{\mathbf{HF}} \textit{ is definable in } \Delta^* + \mathbf{Rec} . \tag{4.4}$$

In particular, *in the formulation* we do not need to mention (in contrast with Sect. 5 devoted to LOGSPACE) any linear ordering in defining graph transformers in FO + LFP. However, *in the proof* for $\mathcal{PTIME}_\gamma$ it plays a crucial role. Also, $\mathcal{AG}$s or $\mathcal{EAG}^{(*)}$s may be equally used as the set Codes here, as well as C may be replaced by $\mathrm{C}^\otimes$ or $\mathrm{C}^\odot$. (Actually, we need only $\Delta + \mathbf{Rec}$ to show mutual definability of all these version of C.) It is because of (N/D)LOGSPACE that we need to be more careful in choosing $\mathrm{C}^\otimes$ or $\mathrm{C}^\odot$. PTIME is less problematic (for **HF**).

[17] Strictly speaking, we must write $\Delta^{*\otimes}$ instead of $\Delta^\otimes$ because $\otimes$ does not cover the full strength of $*$ in Δ^*. However, for simplicity, we will use $\Delta^\otimes$ for $\Delta^{*\otimes}$. Thus, TC is not definable even in $\Delta + \mathbf{Rec}$ because $\mathrm{TC} \notin \mathcal{PTIME}_{\gamma'}$ for some specific (rather artificial) version γ' of γ (cf. [46,49]) such that, moreover, $\mathcal{PTIME}_{\gamma'}$ coincides with the class of operations over **HF** which are definable in an appropriate extension of $\Delta + \mathbf{Rec}$.

5. Main result on capturing LOGSPACE over HF

The following proposition is essential for proving the main result on capturing LOGSPACE over **HF**.[18] Let G_1 and $G_2 \in \mathcal{EAG}_<^{(*)}$ be two arbitrary graphs (each supplemented with canonical linear order $<_i \rightleftharpoons <_{G_i}$ and, possibly, transitive closure $\in_i^*$ of $\in_i \rightleftharpoons \in_{G_i}$, $i = 1, 2$). For any two vertices $x \in |G_1|$ and $y \in |G_2|$ of these graphs let

$$\begin{aligned} x \sim_{1,2} y &\rightleftharpoons \gamma(G_1, x) =_{\mathbf{HF}} \gamma(G_2, y) \ , \\ x \sqsubset_{1,2} y &\rightleftharpoons \gamma(G_1, x) \subset_{\mathbf{HF}} \gamma(G_2, y) \ , \\ x <_{1,2} y &\rightleftharpoons \gamma(G_1, x) <_{\mathbf{HF}} \gamma(G_2, y) \ . \end{aligned}$$

where $=_{\mathbf{HF}}$, $\subset_{\mathbf{HF}}$ and $<_{\mathbf{HF}}$ are, respectively, equality, strong inclusion and lexicographical linear ordering on **HF**.

Proposition 5.1. *The relations $x \sim_{1,2} y$, $x \sqsubset_{1,2} y$ and $x <_{1,2} y$ between vertices of any two graphs G_1 and G_2 in $\mathcal{EAG}_<^*$ (respectively, in $\mathcal{EAG}_<$) are uniformly definable by $\mathrm{FO}^{\odot}$-formulas (respectively, by $\mathrm{FO}^{\otimes}$-formulas[19]).*

Proof. Due to extensionality and acyclicity assumption on graphs G_1 and G_2 we can work with their vertices as with **HF**-sets. Evidently, $x \sim_{1,2} y$ holds iff the corresponding subgraphs "TC($\{x\}$)" of G_1 and "TC($\{y\}$)" of G_2 are isomorphic. We can determine whether the (unique, if any) isomorphism $I_{x,y}$ exists as follows. Define $\mathrm{FO}^{\odot}$-formulas (for $i = 1, 2$):

$$\begin{aligned} &\mathrm{next}_x^i(u, u') \rightleftharpoons u \in_i^* x \ \&\ u' \in_i^* x \ \&\ u <_i u' \ \&\ \neg \exists w \in_i^* x (u <_i w <_i u') \ , \\ &\mathrm{Next}_{x,y}(u, v; u', v') \rightleftharpoons \mathrm{next}_x^1(u, u') \ \&\ \mathrm{next}_y^2(v, v') \ , \\ &I_{x,y}(u, v) \rightleftharpoons \mathrm{Next}_{x,y}^{\odot}(\emptyset, \emptyset; u, v) \ . \end{aligned}$$

Note the crucial role of $\odot$ here. It can be used because Next gives a (partial) mapping $\langle u, v\rangle \mapsto \langle u', v'\rangle$. The formula $I_{x,y}(u, v)$ says that vertices $u \in_1^* x \in |G_1|$ and $v \in_2^* y \in |G_2|$ have the same distance according to their positions from the first element "$\emptyset$" in the restricted to the "transitive closures" of $x \in |G_1|$ and $y \in |G_2|$ linear orderings $<_1$ and $<_2$, respectively.

Then, evidently, $x \sim_{1,2} y$ iff there exists a bijection (isomorphism) between "TC($\{x\}$)" and "TC($\{y\}$)" (actually, given by $I_{x,y}$) which preserves relations $<_1$, $\in_1$ and $<_2$, $\in_2$ restricted to "TC($\{x\}$)" and "TC($\{y\}$)", respectively. Therefore we can define in $\mathrm{FO}^{\odot}$ (omitting subscripts in $\sim_{1,2}$, $\sqsubset_{1,2}$ and $<_{1,2}$):

$$\begin{aligned} x \sim y \rightleftharpoons\ &\forall u \in^* x \exists v \in^* y I_{x,y}(u, v) \ \&\ \forall v \in^* y \exists u \in^* x I_{x,y}(u, v) \ \& \\ &\forall u, u' \in^* x \forall v, v' \in^* y [I_{x,y}(u, v) \ \&\ I_{x,y}(u', v') \Rightarrow \\ &\qquad (u \in_1 u' \iff v \in_2 v') \ \&\ (u \in_1 x \iff v \in_2 y)] \ , \end{aligned}$$

[18] Cf. Cases 2, 5 and 8 in the proof of Theorem 5.4.
[19] More precisely, by $\mathrm{FO}^{*,\odot}$-formulas, with $*$ (or $\otimes$) used only for defining $\in^*$.

$x \sqsubset y \rightleftharpoons x \not\sim y \;\&\; \forall u \in_1 x \exists v \in_2 y (u \sim v)$.

It follows from the axiomatic definition (4.2) of canonical ordering $<$ that (recursively)

$$x < y \iff x \sqsubset y \vee (\exists u, v)(E(x, y; u, v) \;\&\; u < v) \tag{5.1}$$

where

$$\begin{aligned} E(x, y; u, v) \rightleftharpoons\; & x \not\sim y \;\&\; x \not\subset y \;\&\; y \not\subset x \;\&\; u \in_1 x \;\&\; v \in_2 y \;\& \\ & (\forall u' \in_1 x | u' >_1 u)(\exists v' \in_2 y | v' >_2 v)(u' \sim v') \;\& \\ & (\forall v' \in_2 y | v' >_2 v)(\exists u' \in_1 x | u' >_1 u)(u' \sim v') \;\& \\ & u \not\sim v \;. \end{aligned} \tag{5.2}$$

Note, that $E(x, y; u, v)$ determines a partial mapping $\langle x, y\rangle \longmapsto \langle u, v\rangle$ defined for any x, y such that $x \not\sim y$, $x \not\subset y$ and $y \not\subset x$. The uniqueness of $\langle u, v\rangle$ is guaranteed by the conjunct $u \not\sim v$. Indeed, if $x = \{u_1 > u_2 > \ldots\}$ and $y = \{v_1 > v_2 > \ldots\}$ then we can find $\langle u, v\rangle$ as the first pair $\langle u_i, v_i\rangle$ such that $u_i \not\sim v_i$. It should exist because of the conjuncts $x \not\sim y$, $x \not\subset y$ and $y \not\subset x$.

Therefore the *recursion* in (5.1) reduces to *iteration* and we can define explicitly in $\mathrm{FO}^{\odot}$

$$x < y \rightleftharpoons x \sqsubset y \vee \exists u, v (E^{\odot}(x, y; u, v) \;\&\; u \sqsubset v) \;.$$

□

Remark 5.2 (On previous approach). It follows that the classes of IC-$\mathrm{FO}^{\otimes}$- (cf. [34]) and $\mathrm{FO}^{\otimes}$-transformers of $\mathcal{EAG}_{<}$s coincide, what resolves positively questions 5 and 6 from Sect. 13 in [34]. Analogously the classes of IC-$\mathrm{FO}^{\odot}$- and $\mathrm{FO}^{\odot}$-transformers of $\mathcal{EAG}^{*}_{<}$s coincide. Other questions from Sect. 13 in [34] remain open.

Remember, that *any* class of finite pointed graphs $\mathcal{G}$, in particular $\mathcal{G} = \mathcal{EAG}^{(*)}_{<}$, defines a corresponding graph encoding of **HF**-sets with Codes $= \mathcal{G}$. Given any Δ' and class(es) of TRANSFORMERS of graphs from $\mathcal{G}$ to $\mathcal{G}$, we may consider the following abbreviation (introduced in [34])

$$\Delta' \sim_{\mathbf{HF}} \left(\mathcal{G} \xrightarrow[\text{TRANSFORMERS}_2]{\text{TRANSFORMERS}_1} \mathcal{G}\right)$$

of the statement:

Δ'-definable operations over **HF** *coincide with those computable by* TRANSFORMERS$_i$ *for* $i = 1, 2$.

One of the classes of transformers may be omitted in this denotation. If both are present then it is asserted that they compute the *same* class of operations over

HF, namely those Δ'-definable (even if these classes of TRANSFORMERS are different).

Remark 5.3. (Cf. [34].) Fortunately, in the case of $\mathcal{G} = \mathcal{EAG}_<^{(*)}$ we can even check correctness of input and output codes in FO, i.e. the membership relations '$G \in \mathcal{EAG}_<^*$' and '$G \in \mathcal{EAG}_<$'.[20] *Extensionality* and all axioms of a strict linear and canonical ordering $<$ (cf. (4.1) for the latter) are evidently expressible by FO-formulas. Then *acyclicity* of any finite graph G follows from an additional axiom $x \in y \Rightarrow x < y$ which, of course, must hold on $\mathcal{EAG}_<$s. This allows to use $\in$-induction in proving that $<$ is uniquely defined by these axioms on any $\mathcal{EAG}$ and therefore coincides with the *canonical ordering.* Finally, the *transitive closure relation* $\in_G^*$ is completely characterised on all the structures satisfying these axioms by the formula $x \in^* y \iff x \in y \vee \exists z \in y (x \in^* z)$ also by using $\in$-induction (on y).

Finally, we are able to formulate and prove the Main Result of this paper:

Theorem 5.4 (Capturing (N/D)LOGSPACE over **HF**).

$$\Delta_<^{\otimes} + \mathrm{C}^{\otimes} \sim_{\mathbf{HF}} \left(\mathcal{EAG}_< \xrightarrow[\text{NLOGSPACE}]{\text{FO}^{\otimes}} \mathcal{EAG}_<\right) \sim_{\mathbf{HF}} \mathcal{NLOGSPACE}_{\mathcal{EAG}_<} \tag{5.3}$$

$$\Delta_<^{\odot} + \mathrm{C}^{\odot} \sim_{\mathbf{HF}} \left(\mathcal{EAG}_<^* \xrightarrow[\text{DLOGSPACE}]{\text{FO}^{\odot}} \mathcal{EAG}_<^*\right) \sim_{\mathbf{HF}} \mathcal{DLOGSPACE}_{\mathcal{EAG}_<^*} \tag{5.4}$$

Moreover, in both cases any definable set-theoretic operation may be equivalently represented by a term of the form $\mathrm{C}^{\otimes/\odot}(t(\bar{x}))$ with $t(\bar{x})$ involving no collapsing ("external collapsing normal form"), and definable predicates do not need to use collapsing at all.[21]

By comparison this theorem with Theorem 4.3 (and by some additional simple consideration) we can get

Corollary 5.5.

$$\text{PTIME} = \text{(N/D)LOGSPACE} \iff \Delta^* + \mathbf{Rec} + \mathrm{C} \sim_{\mathbf{HF}} \Delta_<^{\otimes/\odot} + \mathrm{C}^{\otimes/\odot}.$$

[20] This will be used in the proof of Theorem 5.4, Case 6.

[21] In [34], the first approach to NLOGSPACE over **HF** (mentioned in Sect. 2) was based on such a syntax with *external* collapsing, and only for such restricted language we were able to obtain a result analogous to (5.3) or (5.4). Now we see that, by using Proposition 5.1 in the proof below, this artificial restriction may be avoided (it proves to be not a restriction at all from the point of view of definability!) and, in particular, the corresponding class of (N/D)LOGSPACE computable operations and predicates is closed under composition.

According to Sect. 3.5 and 3.7, in the proof below we will not use at all logarithmic space *computability* of graph transformers in favour of $\mathrm{FO}^{\otimes/\odot}$-*definability*. Therefore appropriate logical formulas instead of LOGSPACE algorithms will be used.

> In a sense it is just *reducing the considered version of the Δ-language to a version of the relational language.*

Proof of (5.3), *direct part* (NLOGSPACE- or $\mathrm{FO}^{\otimes}$-computability of queries over **HF** which are definable in $\Delta^{\otimes}_{<} + \mathrm{C}^{\otimes}$).

For each term $t(\bar{x}) = t(x_1, \ldots, x_n)$ in $\Delta^{\otimes}_{<} + \mathrm{C}^{\otimes}$ (representing a query q in the diagram (3.4)) we will define in $\mathrm{FO}^{\otimes}$ (in the sense of Sect. 3.5) by induction on the construction of t a corresponding graph transformer

$$[\![t]\!] : \bar{G} = \langle G_1, \ldots, G_n \rangle \longmapsto G_{t(\bar{x})} : (\mathcal{AG}_{=,<})^n \to \mathcal{AG}_{=,<}$$

($[\![t]\!]$ corresponds to Q in terms of (3.4)) with $G_i = \langle |G_i|; =_i, \in_i, <_i, V_i \rangle$. (Cf. agreement in Sect. 3.4, especially (3.6); for the input graphs G_i we may consider that $=_i$ is just the ordinary identity and V_i is a singleton consisting of the distinguished vertex of G_i.) This should be done in such a way that the corresponding version of *the diagram (3.4) will commute — the main statement, alongside with* $\mathrm{FO}^{\otimes}$*-definability, to be proved in the direct part of (5.3).* We left it to the reader to check this commutativity statement and concentrate mainly on the definition in $\mathrm{FO}^{\otimes}$ of $[\![t]\!]$ (or G_t by $\bar{G}$). Everything can be done by straightforward structural induction on the term t. Note, that x_i is simultaneously a set-theoretic variable of the term t and a first-order variable denoting any (or, when necessary, distinguished) vertex of G_i. According to Sect 3.5 and Footnote 8 on using tilde-variables we define $[\![t]\!]$ by a tuple of $\mathrm{FO}^{\otimes}$-formulas

$$G_{t(\bar{x})} = \langle \tilde{u} =_{t(\bar{x})} \tilde{v}, \tilde{u} \in_{t(\bar{x})} \tilde{v}, \tilde{u} <_{t(\bar{x})} \tilde{v}, \tilde{v} \in V_{t(\bar{x})} \rangle \tag{5.5}$$

(depending on tuples of variables $\tilde{u}, \tilde{v}, \bar{x}$) over many-sorted structure $\bar{G} = G_1, \ldots, G_n$. Here it is required that $\bar{G} \models$ '$\tilde{u} =_{t(\bar{x})} \tilde{v}$ is a partial equivalence relation of $\tilde{u}, \tilde{v}$' and that $|G_{t(\bar{x})}| = \{\tilde{u} : \tilde{u} =_{t(\bar{x})} \tilde{u}\}$ and $\bar{G} \models \forall \tilde{u}, \tilde{v}(\tilde{u}, \tilde{v} \in V_{t(\bar{x})} \Rightarrow \tilde{u} =_{t(\bar{x})} \tilde{v})$, and even more; cf. Sect. 3.4, especially (3.6) for the case of G_t, and Sect. 3.5.

We will not guarantee that $=_{t(\bar{x})}$ will be a congruence w.r.t. $\in_{t(\bar{x})}$, and $V_{t(\bar{x})}$, unlike $<_{t(\bar{x})}$ (cf. Footnote 7), but we do guarantee that (after "improvement" of these relations as in Sect. 3.4) and, assuming,

$$G_i \models x_i \in V_i,$$

the quotient $G_t/{=_t}$ is the required graph. Actually, $\tilde{u} =_t \tilde{v}$ will assert that $\tilde{u}$ and $\tilde{v}$ denote the same sets w.r.t. the main graph relation $\in_t$, and similarly for $<_t$.[22]

[22] According to Footnote 9, $=_t$ may be eventually considered to be (partial) identity. Therefore, we can also assume this for G_{t_i}, G_a, G_s, etc. arising in induction assumptions below.

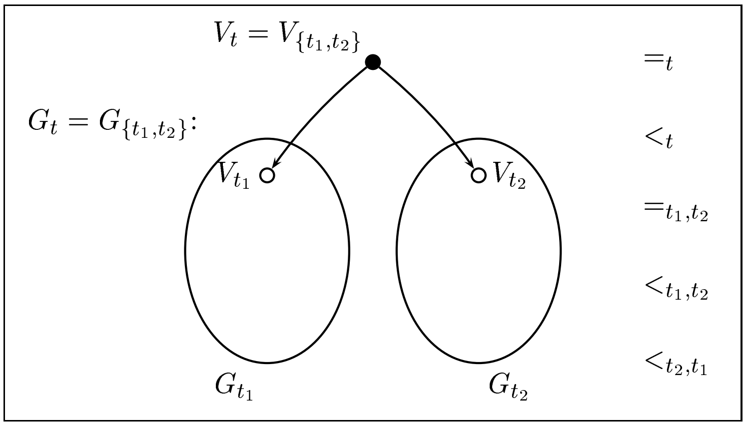

Figure 5.1. Graph G_t for $t = \{t_1, t_2\}$

Analogously, for each formula $\varphi(\bar{x})$ in $\Delta^{\otimes}_{<} + \mathrm{C}^{\otimes}$ we will define by induction on the construction of φ a corresponding $\mathrm{FO}^{\otimes}$-formula $[\![\varphi]\!](\bar{x})$ over many-sorted structure $\bar{G} = G_1, \ldots, G_n$ (depending only on $\bar{x}$ and on no variables of tilde-tuples; the latter variables may be used only as closed in $[\![\varphi]\!](\bar{x})$).

Case **1.** If $t(\bar{x})$ is just a variable x_i then let $G_t \rightleftharpoons G_i$: $u =_{t(\bar{x})} v \rightleftharpoons u =_i v$, $u \in_{t(\bar{x})} v \rightleftharpoons u \in_i v$, $u <_{t(\bar{x})} v \rightleftharpoons u <_i v$, and $v \in V_{t(\bar{x})} \rightleftharpoons v \in V_i$ (for $\tilde{u}$ and $\tilde{v}$ being just u and v, respectively).

Case **2.** Let $t(\bar{x})$ be $\{t_1(\bar{x}), t_2(\bar{x})\}$. By the induction hypothesis, terms $t_1(\bar{x})$ and $t_2(\bar{x})$ have a corresponding two $\mathrm{FO}^{\otimes}$-definable graph transformers $\bar{G} \longmapsto G_{t_1(\bar{x})}$ and $\bar{G} \longmapsto G_{t_2(\bar{x})}$, $G_{t_i(\bar{x})} = \langle u_i =_{t_i} v_i, u_i \in_{t_i} v_i, u_i <_{t_i} v_i, u_i \in V_{t_i}\rangle$ in the language of $\bar{G}$ (with u and v being for simplicity single variables and with variables $\bar{x}$ ranging, respectively, over $\bar{G}$ to denote (in particular) distinguished vertices of these graphs).[23]

The resulting graph $G_t = G_{\{t_1,t_2\}}$ (cf. (5.5) above) is combined from G_{t_1} and G_{t_2} as in the Fig. 5.1). Just one new vertex $V_t = V_{\{t_1,t_2\}}$ and two outgoing edges to V_{t_1} and V_{t_2} are drawn. It is assumed that, by using Proposition 5.1, $=_t$ will identify vertices on this picture which denote the same **HF**-sets. We define also $<_t$ by the same Proposition.

We should imitate this construction by $\mathrm{FO}^{\otimes}$-formulas. To this end, let us abbreviate (for the Case 2 only) $\tilde{u} \rightleftharpoons u_1', u_1, u_2$ and analogously for $\tilde{v}$, $\tilde{w}$, $\tilde{z}$ and $\tilde{h}$. Extend the denotation $u_i \in |G_{t_i}|$ to tuples of variables $\tilde{u}$:

$$\tilde{u} \in |G_{t_1}| \rightleftharpoons u_1' =_{t_1} \emptyset \;\&\; u_1 \in |G_{t_1}| \;\&\; u_2 =_{t_2} \{\emptyset\}\ ,$$

[23] Strictly speaking, u_i and v_i should be tuples of variables $\tilde{u}_i$ and $\tilde{v}_i$. For simplicity, we will mostly use tilde-tuples for the resulting graph G_t. Note, that u_i and v_i range over $|G_{t_i}|$, $i = 1, 2$.

$$\tilde{u} \in |G_{t_2}| \rightleftharpoons u_1' =_{t_1} \{\emptyset\} \ \& \ u_1 =_{t_1} \{\emptyset\} \ \& \ u_2 \in |G_{t_2}| \ ,$$

thereby defining *new versions* of $|G_{t_1}|$ and $|G_{t_2}|$. (For simplicity we do not change the denotation.) Here the new, auxiliary variable u_1' ranges, say, over $|G_{t_1}|$. (Also Remark 4.2 is used here and below for defining in FO the equalities $u = \emptyset$ and $u = \{\emptyset\}$ and other set theoretic formulas.) The mappings

$$u_1 \longmapsto \emptyset, u_1, \{\emptyset\} \text{ and } u_2 \longmapsto \{\emptyset\}, \emptyset, u_2$$

give natural bijections between old and new versions of $|G_{t_1}|$ and $|G_{t_2}|$ in $|\tilde{G}| \rightleftharpoons |G_1| \times |G_1| \times |G_2|$. The latter Cartesian product gives us the "necessary amount of material" from which G_t will be constructed. Also note, that new versions of $|G_{t_1}|$, $|G_{t_2}|$ and the set V_t defined below are mutually disjoint and, moreover, V_t is a singleton representing the distinguished vertex of the resulting graph G_t. (On the role of V_t cf. again Sect. 3.4.)

Define G_t by the following formulas (cf. also Fig. 5.1) where the FO$^{\otimes}$-definitions of $=_{t_1,t_2}$ and $<_{t_1,t_2}$ are based on Proposition 5.1 via $\sim_{1,2}$ and $<_{1,2}$. We use % to give informal comments.

$\tilde{u} \in V_t \rightleftharpoons u_1' =_{t_1} \emptyset \ \& \ u_1 =_{t_1} \emptyset \ \& \ u_2 =_{t_2} \emptyset$,
  % $\langle \emptyset, \emptyset, \emptyset \rangle$ *is the new distinguished vertex (of* G_t*)*
$|G_t| \rightleftharpoons |G_{t_1}| \cup |G_{t_2}| \cup V_t$,
  % V_t *and new version of* $|G_{t_i}|$ *are mutually disjoint subsets of* $|\tilde{G}|$
$\tilde{u} \in_t \tilde{v} \rightleftharpoons (\tilde{u}, \tilde{v} \in |G_{t_1}| \ \& \ u_1 \in_{t_1} v_1) \vee (\tilde{u}, \tilde{v} \in |G_{t_2}| \ \& \ u_2 \in_{t_2} v_2) \vee$
  % *the above disjuncts imitate* $\in_{t_i}$, $i = 1, 2$, *for the embedded* $G_{t_i} \subseteq G_t$
 $(\tilde{u} \in |G_{t_1}| \ \& \ u_1 \in V_{t_1} \ \& \ \tilde{v} \in V_t) \vee (\tilde{u} \in |G_{t_2}| \ \& \ u_2 \in V_{t_2} \ \& \ \tilde{v} \in V_t)$,
  % *this imitates the edges* $V_t \to V_{t_i}$, $i = 1, 2$
$\tilde{u} =_t^0 \tilde{v} \rightleftharpoons (\tilde{u}, \tilde{v} \in |G_{t_1}| \ \& \ u_1 =_{t_1} v_1) \vee (\tilde{u}, \tilde{v} \in |G_{t_2}| \ \& \ u_2 =_{t_2} v_2) \vee$
 $(\tilde{u} \in |G_{t_1}| \ \& \ \tilde{v} \in |G_{t_2}| \ \& \ u_1 =_{t_1,t_2} v_2) \vee$
 $(\tilde{u} \in |G_{t_2}| \ \& \ \tilde{v} \in |G_{t_1}| \ \& \ v_1 =_{t_1,t_2} u_2)$,
  % *this imitates equivalence* $=_t$ *only on* $|G_{t_1}| \cup |G_{t_2}|$
  % *by using Proposition 5.1*
$\tilde{u} =_t \tilde{v} \rightleftharpoons \forall \tilde{z} \in_t \tilde{u} \exists \tilde{w} \in_t \tilde{v} (\tilde{z} =_t^0 \tilde{w}) \ \& \ \forall \tilde{w} \in_t \tilde{v} \exists \tilde{z} \in_t \tilde{u} (\tilde{z} =_t^0 \tilde{w})$,
  % *this extends* $=_t^0$ *to* $|G_t| = |G_{t_1}| \cup |G_{t_2}| \cup V_t$ *by extensionality;*
  % *note that the distinguished vertex in* V_t *may be incidentally*
  % $=_t$ *to a vertex in* G_{t_1} *or* G_{t_2}
$\tilde{u} <_t^0 \tilde{v} \rightleftharpoons (\tilde{u}, \tilde{v} \in |G_{t_1}| \ \& \ u_1 <_{t_1} v_1) \vee (\tilde{u}, \tilde{v} \in |G_{t_2}| \ \& \ u_2 <_{t_2} v_2) \vee$
 $(\tilde{u} \in |G_{t_1}| \ \& \ \tilde{v} \in |G_{t_2}| \ \& \ u_1 <_{t_1,t_2} v_2) \vee$
 $(\tilde{u} \in |G_{t_2}| \ \& \ \tilde{v} \in |G_{t_1}| \ \& \ u_2 <_{t_2,t_1} v_1)$,
  % *this imitates* $<_t$ *only on* $|G_{t_1}| \cup |G_{t_2}|$ *by using Proposition 5.1*

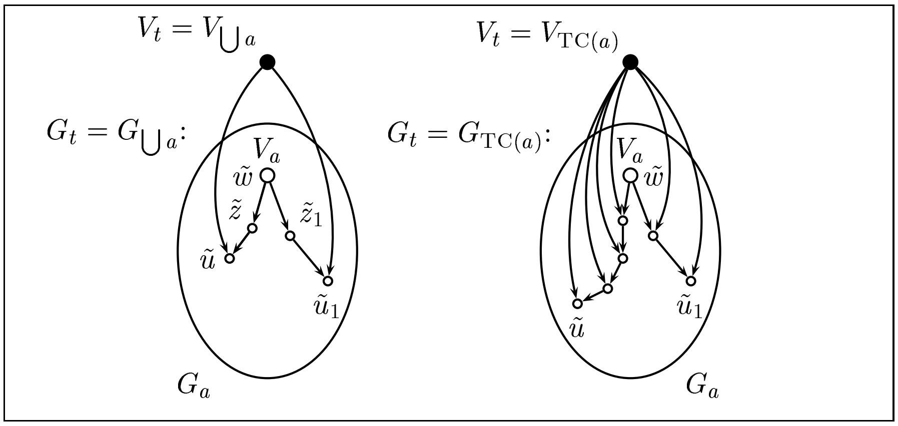

Figure 5.2. Graphs G_t for $t = \bigcup a$ and for $t = \mathrm{TC}(a)$

$\tilde{u} <_t \tilde{v} \rightleftharpoons (\exists \tilde{z} \in_t \tilde{v} \setminus \tilde{u})(\forall \tilde{w} \in_t \tilde{u} \cup \tilde{v} | \tilde{z} <_t^0 \tilde{w})(\exists \tilde{h} \in_t \tilde{u} \cap \tilde{v})(\tilde{w} =_t^0 \tilde{h})$.
% *extends* $<_t^0$ *to* $|G_t|$ *according to* (4.1)

Here we used the following abbreviations of quantifier prefixes:

$$(\exists \tilde{z} \in_t \tilde{v} \setminus \tilde{u}) \rightleftharpoons (\exists \tilde{z} \in_t \tilde{v} | \forall \tilde{z}' \in \tilde{u}(\tilde{z} \neq_t^0 \tilde{z}')) ,$$
$$(\forall \tilde{w} \in_t \tilde{u} \cup \tilde{v} | \tilde{z} <_t^0 \tilde{w}) \rightleftharpoons (\forall \tilde{w} | (\tilde{w} \in_t \tilde{u} \vee \tilde{w} \in_t \tilde{v}) \ \&\ \tilde{z} <_t^0 \tilde{w}))$$

and analogously for $(\exists \tilde{h} \in_t \tilde{u} \cap \tilde{v})$. Note, that $=_t$ should coincide with $=_t^0$ on $|G_{t_1}| \cup |G_{t_2}|$, and the same for $<_t$ and $<_t^0$.

Case **3.** Let $t(\bar{x})$ be $\bigcup a(\bar{x})$. By the induction hypothesis, the term a has a corresponding FO$^{\otimes}$-definable graph transformer $\bar{G} \longmapsto G_{a(\bar{x})}$,

$$G_{a(\bar{x})} = \langle u =_a v, u \in_a v, u <_a v, u \in V_a \rangle$$

in the language of $\bar{G}$ (with u and v being for simplicity single variables and with variables $\bar{x}$ ranging, respectively, over $\bar{G}$ to denote (arbitrary or distinguished) vertices of these graphs).

The resulting graph G_t is constructed from G_a as in Fig. 5.2 (left part). Just one new vertex $V_t = V_{\bigcup a}$ and many outgoing edges to "elements of elements" ("grandchildren") of the vertex (in) V_a are drawn.

To imitate this in FO$^{\otimes}$, first abbreviate (for the cases 3 and 4 only) $\tilde{u} \rightleftharpoons u', u$, and analogously for $\tilde{v}, \tilde{w}, \tilde{h}, \ldots$, and extend the denotation $u \in |G_a|$ to tuples of variables $\tilde{u}$:

$$\tilde{u} \in |G_a| \rightleftharpoons u' =_a \{\emptyset\} \ \&\ u \in |G_a|$$

thereby defining a *new version* of $|G_a|$. Define G_t by the following formulas:

$$\begin{aligned}
&\tilde{v} \in V_t \rightleftharpoons v' =_a \emptyset \;\&\; v =_a \emptyset\ ,\\
&|G_t| \rightleftharpoons |G_a| \cup V_t\ ,\\
&\tilde{u} \in_t \tilde{v} \rightleftharpoons (\tilde{u}, \tilde{v} \in |G_a| \;\&\; u \in_a v) \vee\\
&\qquad (\tilde{u} \in |G_a| \;\&\; \tilde{v} \in V_t \;\&\; \exists \tilde{z} \in |G_a| \exists \tilde{w} \in V_a (u \in_a z \in_a w))\ ,\\
&\tilde{u} =_t \tilde{v} \rightleftharpoons \forall \tilde{z} \in_t \tilde{u} \exists \tilde{w} \in_t \tilde{v} (z =_a w) \;\&\; \forall \tilde{w} \in_t \tilde{v} \exists \tilde{z} \in_t \tilde{u} (z =_a w)\ ,\\
&\tilde{u} <_t \tilde{v} \rightleftharpoons (\exists \tilde{z} \in_t \tilde{v} \setminus \tilde{u})(\forall \tilde{w} \in_t \tilde{u} \cup \tilde{v} | z <_a w)(\exists \tilde{h} \in_t \tilde{u} \cap \tilde{v})(w =_a h)\ .
\end{aligned}$$

It is worth to recall that $\tilde{w} \rightleftharpoons w', w$, etc.

Case **4.** Let $t(\bar{x})$ be $\mathrm{TC}(a(\bar{x}))$ (cf. Fig. 5.2, right part). This differs from Case 3 only by the definition

$$\tilde{u} \in_t \tilde{v} \rightleftharpoons (\tilde{u}, \tilde{v} \in |G_a| \;\&\; u \in_a v) \vee (\tilde{u} \in |G_a| \;\&\; \tilde{v} \in V_t \;\&\; \exists \tilde{w} \in V_a (u \in_a^{\otimes} w))\ .$$

Case **5.** Let $t(\bar{x})$ be $\{s(\bar{x}, y) : y \in a(\bar{x}) \;\&\; \varphi(\bar{x}, y)\}$. By the induction hypothesis, there are $\mathrm{FO}^{\otimes}$-definable graph transformers $[\![s]\!] : \bar{G}, G_0 \longmapsto G_{s(\bar{x}, x_0)}$, $[\![a]\!] : \bar{G} \longmapsto G_{a(\bar{x})}$ and a predicate $[\![\varphi]\!]$ over $\bar{G}, G_0$ where

$$G_{s(\bar{x},x_0)} = \langle u =_{s(\bar{x},x_0)} v, u \in_{s(\bar{x},x_0)} v, u <_{s(\bar{x},x_0)} v, v \in V_{s(\bar{x},x_0)} \rangle$$

is defined in terms of $\bar{G}, G_0$ (with u and v being for simplicity single variables; strictly speaking they should be tuples ranging someway over $\bar{G}, G_0$) and

$$G_{a(\bar{x})} = \langle \tilde{y} =_{a(\bar{x})} \tilde{z}, \tilde{y} \in_{a(\bar{x})} \tilde{z}, \tilde{y} <_{a(\bar{x})} \tilde{z}, \tilde{y} \in V_{a(\bar{x})} \rangle$$

is defined in terms of $\bar{G}$ (with variables $\bar{x}, x_0$ ranging, respectively, over $\bar{G}, G_0$ to denote any (in particular, distinguished) vertices of these graphs).

Informally, the resulting graph G_t is combined from the family of graphs $G_{s(\bar{x},y)}$ for $y \in a$ (cf. more precise definition below) according to Fig. 5.3 (in which φ is omitted for simplicity).

Note, that formulas defining $G_{s(\bar{x},x_0)}$ should involve, in particular, the predicate symbols $=_0, \in_0, <_0$ and V_0 from the structure G_0. As, *set-theoretically*, y (corresponding to the variable x_0) should range over the set $a(\bar{x})$ and $a(\bar{x})$ is represented by the distinguished vertex in $V_{a(\bar{x})}$ of $G_{a(\bar{x})}$ we, therefore, should just take G_a in the role of G_0. To this end, let us rewrite $G_{s(\bar{x},x_0)}$ in the language of $\bar{G}$ only (without using G_0) by substituting $G_{a(\bar{x})} = \langle \tilde{y} =_{a(\bar{x})} \tilde{z}, \tilde{y} \in_{a(\bar{x})} \tilde{z}, \tilde{y} <_{a(\bar{x})} \tilde{z}, \tilde{y} \in V_{a(\bar{x})} \rangle$ in place of $G_0 = \langle |G_0|, =_0, \in_0, <_0, V_0 \rangle$ (i.e. by substituting the formulas for $=_{a(\bar{x})}$, $\in_{a(\bar{x})}$, $<_{a(\bar{x})}$, $V_{a(\bar{x})}$, respectively, in place of the predicates $=_0$, $\in_0$, $<_0$ and V_0 and with the variables $\tilde{y}$, $\tilde{z}$, etc. ranging over $|G_{a(\bar{x})}|$ and playing the role of variables y, z, etc. ranging over $|G_0|$, in particular, x_0 will become $\tilde{x}_0$;

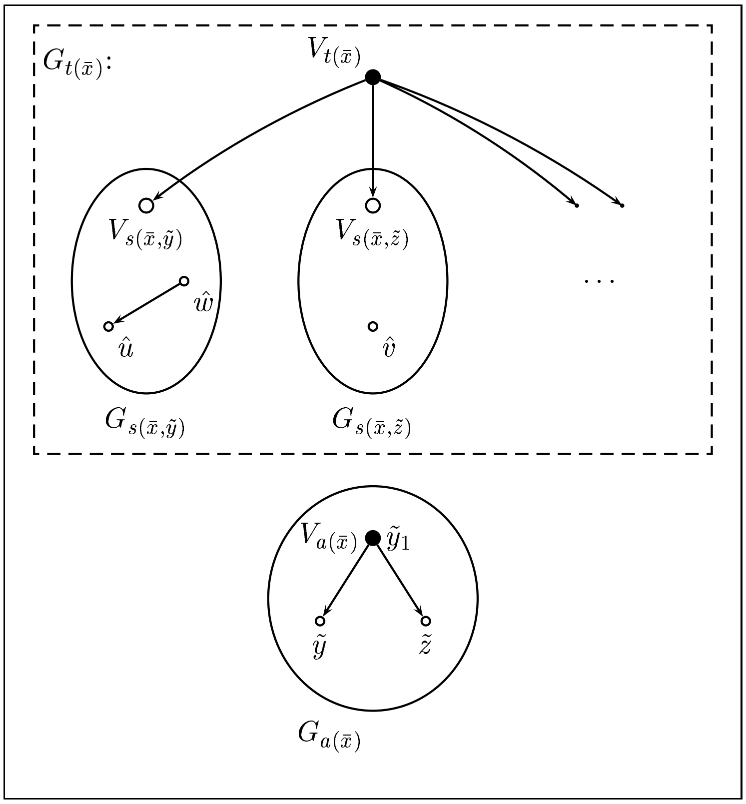

Figure 5.3. Graph G_t for $t(\bar{x}) = \{s(\bar{x}, y) : y \in a(\bar{x})\}$

variables $\bar{x} = x_1, \ldots, x_n$ are still ranging over $|\bar{G}| = |G_1|, \ldots, |G_n|$). Denote the result as

$$G_{s(\bar{x},\tilde{x}_0)} = \langle u =_{s(\bar{x},\tilde{x}_0)} v, u \in_{s(\bar{x},\tilde{x}_0)} v, u <_{s(\bar{x},\tilde{x}_0)} v, v \in V_{t(\tilde{x}_0)} \rangle$$

(almost in the same way as the original $G_{s(\bar{x},x_0)}$; only $\tilde{x}_0$ witnesses the change; however, strictly speaking, also u and v, which actually may be tuples of variables, may involve now variables like $\tilde{y}, \tilde{z}$ above if before substituting G_a for G_0 they involved variables ranging over G_0). Analogously $[\![\varphi]\!](\bar{x}, \tilde{x}_0)$ is obtained from $[\![\varphi]\!](\bar{x}, x_0)$.

Now, let us substitute all the children $\tilde{y}, \tilde{z}, \ldots \in \tilde{y}_1 \in V_a$ of the distinguished vertex of G_a for the variable(s) $\tilde{x}_0$. Then $V_{s(\bar{x},\tilde{y})}, V_{s(\bar{x},\tilde{z})}, \ldots$ denote in the graphs $G_{s(\bar{x},\tilde{y})}, G_{s(\bar{x},\tilde{z})}, \ldots$ the required sets $s(\bar{x}, y), s(\bar{x}, z), \ldots$ for $y, z, \ldots \in a(\bar{x})$ and the new vertex $V_{t(\bar{x})}$ with the new edges $V_{t(\bar{x})} \to V_{s(\bar{x},\tilde{y})}$, $V_{t(\bar{x})} \to V_{s(\bar{x},\tilde{z})}$, etc. will denote the required set $\{s(\bar{x}, y) : y \in a(\bar{x})\}$ (with the restriction $\varphi(\bar{x}, y)$ omitted for simplicity). This is, roughly, the way the resulting graph G_t is obtained. We should additionally care of FO$^{\otimes}$-definability of the required graph and of the corresponding equality $=_t$ and canonical linear ordering $<_t$ predicates.

Abbreviate $\hat{u} \rightleftharpoons u, \tilde{y}, \tilde{y}'$, $\hat{v} \rightleftharpoons v, \tilde{z}, \tilde{z}'$, $\hat{w} \rightleftharpoons w, \tilde{p}, \tilde{p}'$, $\hat{h} \rightleftharpoons h, \tilde{q}, \tilde{q}'$ (cf. Footnote 8) and define G_t by the following formulas (cf. also Fig. 5.3):

$\hat{u} \in |G^0_{t(\bar{x})}| \rightleftharpoons u \in |G_{s(\bar{x},\tilde{y})}|$ & $\tilde{y} \in |G_{a(\bar{x})}|$ & $\tilde{y}' =_{a(\bar{x})} \{\emptyset\}$ &

% *disjoint union* $\sum_{\tilde{y} \in |G_{a(\bar{x})}|} |G_{s(\bar{x},\tilde{y})}|$

$\exists \tilde{y}_1 \in V_{a(\bar{x})}(\tilde{y} \in_{a(\bar{x})} \tilde{y}_1)$ & $[\![\varphi]\!](\bar{x}, \tilde{y})$,

% $V_{a(\bar{x})} \ni \tilde{y}_1 \to \tilde{y}$ *should exist in* $G_{a(\bar{x})}$ *(as in Fig. 5.3)*

% *with additional restriction* φ *on* $\tilde{y}$

$\hat{u} \in V_{t(\bar{x})} \rightleftharpoons u =_{s(\bar{x},\tilde{y})} \emptyset$ & $\tilde{y} =_{a(\bar{x})} \emptyset$ & $\tilde{y}' =_{a(\bar{x})} \emptyset$,

% $\langle \emptyset, \emptyset, \emptyset \rangle$ *is the new distinguished vertex (of* G_t*)*

$|G_{t(\bar{x})}| \rightleftharpoons |G^0_{t(\bar{x})}| \cup V_{t(\bar{x})}$, % *disjoint union*

$\hat{u} \in^0_{t(\bar{x})} \hat{w} \rightleftharpoons \hat{u}, \hat{w} \in |G^0_{t(\bar{x})}|$ & $\tilde{y} =_{a(\bar{x})} \tilde{p}$ & $u \in_{s(\bar{x},\tilde{y})} w$,

% $\in_t$ *as it was originally in each disjoint addendum* $(\tilde{y} =_a \tilde{p})$*;*

% *(recall that* $\hat{u} \rightleftharpoons u, \tilde{y}, \tilde{y}'$ *and* $\hat{w} \rightleftharpoons w, \tilde{p}, \tilde{p}'$*)*;

% *cf. the edge* $\hat{w} \to \hat{u}$ *in Fig. 5.3*

$\hat{u} \in_{t(\bar{x})} \hat{w} \rightleftharpoons \hat{u} \in^0_{t(\bar{x})} \hat{w} \vee (\hat{u} \in |G^0_{t(\bar{x})}|$ & $u \in V_{s(\bar{x},\tilde{y})}$ & $\hat{w} \in V_{t(\bar{x})})$,

% *plus new edges of the kind* $V_{t(\bar{x})} \to V_{s(\bar{x},\tilde{y})}$

% *(recall that* $\hat{u} \rightleftharpoons u, \tilde{y}, \tilde{y}'$*)*

$\hat{u} =^0_{t(\bar{x})} \hat{v} \rightleftharpoons \hat{u}, \hat{v} \in |G^0_{t(\bar{x})}|$ & $u =_{s(\bar{x},\tilde{y}),s(\bar{x},\tilde{z})} v$,

% *in the disjoint sum* $|G^0_{t(\bar{x})}|$ *by Proposition 5.1*

% *(recall that* $\hat{u} \rightleftharpoons u, \tilde{y}, \tilde{y}'$ *and* $\hat{v} \rightleftharpoons v, \tilde{z}, \tilde{z}'$*)*

$\hat{u} =_t \hat{v} \rightleftharpoons \forall \hat{z} \in_t \hat{u} \exists \hat{w} \in_t \hat{v} (\hat{z} =^0_t \hat{w})$ & $\forall \hat{w} \in_t \hat{v} \exists \hat{z} \in_t \hat{u} (\hat{z} =^0_t \hat{w})$,

% *in the whole* $|G_{t(\bar{x})}|$ *by extensionality*

$\hat{u} <^0_{t(\bar{x})} \hat{v} \rightleftharpoons \hat{u}, \hat{v} \in |G^0_{t(\bar{x})}|$ & $u <_{s(\bar{x},\tilde{y}),s(\bar{x},\tilde{z})} v$,

% *by Proposition 5.1*

$\hat{u} <_t \hat{v} \rightleftharpoons (\exists \hat{z} \in_t \hat{v} \setminus \hat{u})(\forall \hat{w} \in_t \hat{u} \cup \hat{v} | \hat{z} <^0_t \hat{w})(\exists \hat{h} \in_t \hat{u} \cap \hat{v})(\hat{w} =^0_t \hat{h})$.

% *by (4.1)*

Case **6.** Let $t(\bar{x})$ be $\mathrm{C}^{\otimes}(a(\bar{x}))$. By the induction hypothesis, the term a has the corresponding $\mathrm{FO}^{\otimes}$-definable graph transformer $\bar{G} \longmapsto G_{a(\bar{x})}$,

$$G_{a(\bar{x})} = \langle u =_a v, u \in_a v, u <_a v, u \in V_a \rangle \ .$$

First note, that it easily follows from Remark 5.3 that there exists also an FO-definable predicate $\mathcal{EAG}_<(z)$ which, given a vertex z of a graph (in our case of the graph $G_{a(\bar{x})}$ in $\mathcal{EAG}_<$), decides whether z considered as an **HF**-set corresponds to a triple $z = \langle g, l, p \rangle$ such that (i) g (as an **HF**-set of ordered pairs $\langle u, v \rangle$) is an extensional acyclic graph, (ii) l (also as an **HF**-set of ordered pairs) is

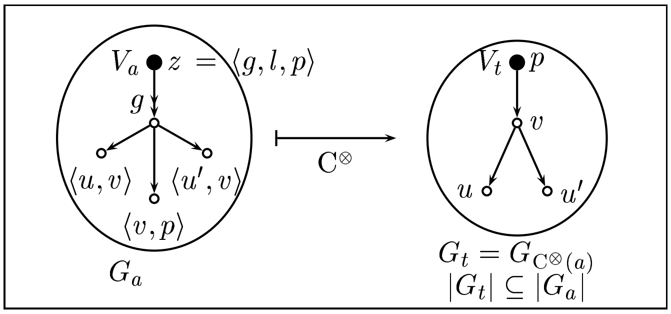

Figure 5.4. Graph G_t for $t = \mathrm{C}^{\otimes}(a)$ ($\mathrm{C}^{\otimes}$ as graph transformer)

the canonical linear ordering on this graph, and, finally, (iii) p is a distinguished vertex of the graph g (possibly isolated one)— a trivial, always true property.

Then, if $z \in V_a$ is the distinguished vertex of G_a, the resulting graph G_t shown in the Fig. 5.4 is the graph discussed in the previous paragraph. We could say informally that the "horizontal" graph given by z in G_a is simply turned into a "vertical" position G_t. We just think on ordered pairs $\langle u, v\rangle$ as on "horizontal" arrows $v \to u$ and on the membership relation arrows as on (more or less) vertical, downwards directed ones $\begin{array}{c} v \\ \downarrow \\ u \end{array}$ (as in all the presented figures).

Respectively, define G_t by the following formulas:

$$
\begin{array}{l}
u \in |G_t| \rightleftharpoons (\exists z \in V_a \exists glpv \in |G_a| \\
\qquad (\mathcal{EAG}_<(z) \;\&\; z =_a \langle g, l, p\rangle \;\&\; (\langle u, v\rangle \in g \vee \langle v, u\rangle \in g \vee u =_a p))) \vee \\
\qquad\qquad \%\ |G_t| \subseteq |G_a| \textit{ consists of the } \mathbf{field}(g) \textit{ plus } p \\
\qquad (u \in V_a \;\&\; \neg\mathcal{EAG}_<(u)) \ , \\
\qquad\qquad \%\ \textit{exceptional and trivial case of } |G_t| = V_a \\
\qquad\qquad \%\ \textit{consisting of only one vertex in } V_a \\
\qquad\qquad \%\ \textit{if the distinguished vertex of } G_a \textit{ ``is not'' } \mathcal{EAG}_< \\
u \in V_t \rightleftharpoons (\exists z \in V_a \exists glp \in |G_a| \\
\qquad (\mathcal{EAG}_<(z) \;\&\; z =_a \langle g, l, p\rangle \;\&\; u =_a p)) \vee (u \in V_a \;\&\; \neg\mathcal{EAG}_<(u)) \ , \\
\qquad\qquad \%\ p \textit{ is the distinguished vertex of } G_a \\
\qquad\qquad \%\ \textit{(except the abovementioned trivial case)} \\
u \in_t v \rightleftharpoons \exists z \in V_a \exists glp \in |G_a| (\mathcal{EAG}_<(z) \;\&\; z =_a \langle g, l, p\rangle \;\&\; \langle u, v\rangle \in g) \ , \\
\qquad\qquad \%\ \textit{from ``horizontal'' in } g \textit{ to ``vertical'' in } G_t \\
u =_t v \rightleftharpoons u, v \in |G_t| \;\&\; u =_a v \ , \\
\qquad\qquad \%\ \textit{equality on } |G_t| \subseteq |G_a| \textit{ is induced by } G_a \\
u <_t v \rightleftharpoons \exists z \in V_a \exists glp \in |G_a| (\mathcal{EAG}_<(z) \;\&\; z =_a \langle g, l, p\rangle \;\&\; \langle u, v\rangle \in l) \ ,
\end{array}
$$

$\%\ l\ from\ z =_a \langle g, l, p\rangle\ generates\ <_t$

Case **7.** of $t \rightleftharpoons r^{\otimes}$ relies on the construct $_{\otimes}$ of FO$^{\otimes}$. it is somewhat analogous to the Case 6, both dealing with sets of ordered pairs. We only note, that, as G_r may have not enough vertices, we should take $|G_t| \subseteq |G_r|^3$.

Case **8.** of $\varphi \rightleftharpoons t_1 = t_2$ or $t_1 < t_2$ is based immediately on Proposition 5.1.

Case **9.** of $\varphi \rightleftharpoons t_1 \in t_2$, as well as that of bounded quantification, is reducible to those considered:

$$\exists x \in t\varphi(x) \iff \{x \in t : \varphi(x)\} \neq \emptyset\ ,$$
$$t_1 \in t_2 \iff \exists x \in t_2(x = t_1)\ .$$

Case **10.** of the term $\emptyset$ and of boolean connectives is trivial.

Proof of (5.3), *converse part* (definability in $\Delta^{\otimes}_{<} + \mathrm{C}^{\otimes}$ of FO$^{\otimes}$-computable queries over **HF**), and *"external collapsing normal form"*. Let $f(\bar{x})$ be any n-ary operation over sets FO$^{\otimes}$-computable by a graph transformer $F : \mathcal{EAG}_< \to \mathcal{EAG}_<$. We may consider that $\mathcal{EAG}_< \subseteq \mathbf{HF}$ and $F : \mathbf{HF} \to \mathbf{HF}$. Then f can be represented as the following $(\Delta^{\otimes}_{<} + \mathrm{C}^{\otimes})$-definable operation, in fact a composition (applied from right to left)

$$f = \mathrm{C}^{\otimes} \circ F \circ G$$

where G is an n-ary $\Delta^{*}_{<}$-definable operation over sets transforming input sets $x_1, \ldots, x_n$ into their well-founded extensional graph representation (with the distinguished vertex in each graph):

$$G(x_1, \ldots, x_n) = \langle \langle \in\restriction_{\mathrm{TC}(\{x_1\})}; <\restriction_{\mathrm{TC}(\{x_1\})}, x_1\rangle, \ldots, \langle \in\restriction_{\mathrm{TC}(\{x_n\})}; <\restriction_{\mathrm{TC}(\{x_n\})}, x_n\rangle\rangle\ .$$

Δ^*-definition of TC and the tuple construct $\langle \ldots \rangle$ were given in Sect. 4.1. Define $\in\restriction_T\ \rightleftharpoons \{\langle u, v\rangle : u, v \in T\ \&\ u \in v\}$, and analogously for $<\restriction_T$.

F is evidently also $\Delta^{\otimes}$-definable by imitating in $\Delta^{\otimes}$ its FO$^{\otimes}$-description. (The input graph for F is considered as a set of a specific kind; corresponding output graph is defined by using a Δ-definable notion of Cartesian products (Sect. 4.1) and by imitating unbounded quantifiers of FO$^{\otimes}$ and the logical construct $_{\otimes}$ by bounded quantifiers and the set-theoretic version of $_{\otimes}$ in $\Delta^{\otimes}$.) Finally, the collapsing operation $\mathrm{C}^{\otimes}$ serves to 'extract' a resulting set from its graph representation.

Evidently, F and G do not use collapsing. If $f : \mathbf{HF}^n \to \{\mathbf{true,false}\}$ is an NLOGSPACE computable predicate then we will have $f = F \circ G$ and we do not need to use collapsing at all.

Proof of (5.4) essentially coincides with that of (5.3). We should only replace (i) $_{\otimes}$ by $_{\odot}$, (ii) $\mathrm{C}^{\otimes}$ by $\mathrm{C}^{\odot}$, (iii) $\mathcal{EAG}_<$ by $\mathcal{EAG}^*_<$ and add (i$'$) $\in^*_i$ in the denotation of

G_i, (ii′) $\tilde{u} \in^*_{t(\bar{x})} \tilde{v}$ in the denotation of $G_{t(\bar{x})}$ for each term t, (iii′) $\in^* \lceil_{\mathrm{TC}(\{x_i\})}$ in the definition of $G(\bar{x})$, and extend the proof, respectively, with using corresponding version of Proposition 5.1 for $\mathcal{EAG}^*_<$.

□

6. Conclusion

After previous attempts [34] we presented in this paper a sufficiently satisfactory approach to (N/D)LOGSPACE computability over the universe of hereditarily-finite sets **HF** and set-theoretic query languages $\Delta^{\otimes/\odot}_< + \mathrm{C}^{\otimes/\odot}$ which capture these computability notions. The crucial point was a partial definability result (Proposition 5.1) of the canonical (lexicographical) linear ordering on finite extensional acyclic graphs.

Further investigations in this direction could be related to the following questions most of which concern with the well known issue of the linear order on finite structures in descriptive complexity:

1. *Open questions from Sect. 13 in [34] (except 5 and 6 which were resolved in this paper). Specifically,*
   (a) *Is the canonical linear ordering on $\mathcal{EAG}_\prec$s (uniformly) definable in* $\mathrm{FO}^\otimes$*?*
   (b) *Is any linear ordering on $\mathcal{EAG}$s (uniformly) definable in* $\mathrm{FO}^{\otimes/\odot}$*?*
   (c) *Are the classes of* $\mathrm{FO}^{\otimes/\odot}$*-definable and (N/D)LOGSPACE computable queries over $\mathcal{EAG}$s the same?*
   (d) *Is* $\in^*_G$ $\mathrm{FO}^\odot$*-definable on $\mathcal{EAG}_\prec$s?*
2. *Can Proposition 5.1 be strengthened in any way (say, by replacing $\mathcal{EAG}^*_<$ by $\mathcal{EAG}_<$ or $\mathcal{EAG}^*$ or $\mathcal{EAG}$)?*
3. *Is it really inevitable considering the above enriched classes of graphs $\mathcal{EAG}_<$ or $\mathcal{EAG}^*_<$ as codes of* **HF***-sets for describing (N/D)LOGSPACE?*
4. *Is $<_{\mathbf{HF}}$ definable in $\Delta^\otimes$?*
5. *Consider (N/D)LOGSPACE over* **HF** *w.r.t. encoding by trees or bracket expressions (cf. Footnote 3).*
6. *Extend the notion of polynomial regular encoding of* **HF** *to the case of logarithmic space regular encoding (cf. Footnote 3).*
7. *Generalise the results of this paper to the case of (cyclic) hereditarily-finite hyper-sets (***HFA***, cf. [35,36]). In particular, reconsider corresponding computability in terms of Web-like databases (i.e. in terms of browsing, searching and composing of new Web-pages [36]).*

Let us comment briefly on the problem 1 (c) because it might remind such well-known problem of descriptive complexity theory (with widely believable negative answer for general finite structures, say, for *arbitrary* finite graphs) as [13,22]

- *Whether all (generic) PTIME-computable queries over finite first-order structures (of a given signature) can be captured by some logic?*

In our case we could also ask the same question for the case of (N/D)LOGSPACE and $\mathcal{EAG}$. But the latter class of structures is rather special. However it seems doubtful, the problem 1 (b) can, in principle, have a positive solution and therefore imply 1 (c). For example, for the case of $\mathcal{EAG}$s and also for a larger class $\mathcal{SEG}$ of *strongly extensional graphs* (possibly with cycles) a uniform linear ordering is definable in FO + LFP and, therefore, the latter language captures PTIME over such graphs [35,40]. Note, that it was also not very expected that Proposition 5.1, giving in a sense a partial answer to 1 (b), will be proved. Of course, if it would be PTIME=(N/D)LOGSPACE then we would have positive answer to 1 (a)/(d), respectively.

While the answers to the questions 1–4 seem to be difficult and most probably negative, the questions 5–7 look rather as the normal programs for research.

Acknowledgements

The authors are very grateful to anonymous referees for a lot of useful notes, in particular, for the help to improve English language of the paper.

References

[1] S. Abiteboul, Querying semi-structured data. Database Theory — ICDT'97, 6th International Conference, Delphi, Greece, January 1997, Proceedings. Lecture Notes in Computer Science **1186** Springer (1997), 1–18.

[2] S. Abiteboul and C. Beeri, On the power of languages for the manipulation of complex objects. INRIA research report **846** (1988). Abstract in Proc. International Workshop on Theory and Applications of Nested Relations and Complex Objects. Darmstadt (1987).

[3] S.Abiteboul, P.Buneman, D.Suciu, Data on the Web: From Relations to Semistructured Data and XML, Morgan Kaufman Publishers, San Francisco, California, 2000.

[4] S. Abiteboul, R. Hull and V. Vianu, Foundations of Databases. Addison-Wesley, Reading-Massachusetts, 1995.

[5] S. Abiteboul, M Vardi and V. Vianu, Fixpoint Logics, Relational Machines, and Computational Complexity, JACM 44(1), 30–56 (1997).

[6] S. Abiteboul and V. Vianu, Queries and computation on the Web. Database Theory — ICDT'97, 6th International Conference, Delphi, Greece, January 1997, Proceedings. Lecture Notes in Computer Science **1186** Springer (1997), 262–275.

[7] P. Aczel, Non-Well-Founded Sets. CSLI Lecture Notes. No **14** (1988).

[8] J.K. Barwise, Admissible Sets and Structures. Springer, Berlin, 1975.

[9] P. Buneman, S. Davidson, G. Hillebrand and D. Suciu, A query Language and Optimization Techniques for Unstructured Data. Proc. of SIGMOD, San Diego, 1996.

[10] P. Buneman, S. Davidson, M. Fernandez and D. Suciu, Adding structure to unstructured data, — ICDT'97, 6th International Conference, Delphi, Greece, January 1997, Proceedings. Lecture Notes in Computer Science **1186** Springer (1997), pp. 336–350.

[11] A. Blass, Y. Gurevich and S. Shelah Choiceless Polynomial Time Annals of Pure and Applied Logic 100 (1999), 141–187.

[12] Val Breazu-Tannen and R. Subrahmanyam, Logical and computational aspects of programming with sets/bags/lists, a manuscript, 1991.

[13] A. Chandra and D. Harel, Structure and Complexity of Relational Queries, J. Comput. and System Sciences 25 (1982), 99–128.

[14] E. Dahlhaus, Is SETL a suitable language for parallel programming — a theoretical approach. Börger, E., Kleine Buning, H., Richter, M.M. ed. CSL'87. LNCS **329** (1987), 56–63.

[15] E. Dahlhaus and J. Makowsky, The Choice of programming Primitives in SETL-like Languages. ESOP'86. LNCS **213** (1986), 160–172.

[16] E. Dahlhaus and J. Makowsky, Query languages for hierarchic databases. Information and Computation **101** (1992), 1–32.

[17] R.O. Gandy, Set theoretic functions for elementary syntax. Proc. Symp. in Pure Math. Vol. **13**, Part II (1974), 103–126.

[18] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H.Freeman and Company, New York, 1979.

[19] E. Grädel and M. Spielmann, Logspace Reducibility via Abstract State Machines, In Proceedings of World Congress on Formal Methods (FM '99), volume 1709 of Lecture Notes in Computer Science, Springer Verlag, (1999), 1738-1757.

[20] S. Grumbach and V. Vianu, Tractable query languages for complex object databases. Rapports de Recherche N **1573**. INRIA. 1991.

[21] Y. Gurevich, Algebras of feasible functions. FOCS **24** (1983), 210–214.

[22] Y. Gurevich, Logic and Challenge of Computer Science, In: Current Trends in Theoretical Computer Science (ed. E. Börger), Computer Science Press, 1988, 1–57.

[23] Y. Gurevich, Evolving Algebra 1993: Lipari Guide in: Specification and Validation Methods, Ed. E. Brger, Oxford University Press, 1995, 9–36.

[24] Y. Gurevich, May 1997 Draft of the ASM Guide (The Michigan Guide) Tech Report CSE-TR-336-97, EECS Dept, University of Michigan, 1997

[25] N. Immerman, 'Relational queries computable in polynomial time', in: Proc. 14th. ACM Symp. on Theory of Computing, (1982) 147–152; cf. also Inform. and Control **68** (1986), 86–104.

[26] N. Immerman, Languages which captures complexity classes. SIAM J. Comput., **16** 4 (1987), 760–778.

[27] N.Immerman, Descriptive and computational complexity. Proceedings of Symposia in Applied Mathematics, **38** (1989), 75–91.

[28] N. Immerman, S. Patnaik and D. Stemple, The expressiveness of a family of finite set languages. Theoretical Computer Science, **155**, N 1 (1996), 111–140.

[29] R.B. Jensen, The fine structure of the constructible hierarchy. Ann. Math. Logic **4** (1972), 229–308.

[30] A. Kosky, Observational properties of databases with object identity. Technical Report MS-CIS-95-20. Dept. of Computer and Information Science, University of Pennsylvania (1995).

[31] G.M. Kuper and M.Y. Vardi, A new approach to database logic. Proc. 3rd ACM Symp. on Principles of Database Systems 1984.

[32] A.V. Leontjev and V.Yu. Sazonov, Capturing LOGSPACE over Hereditarily-Finite Sets,

in: K.-D. Schewe and B. Thalheim (Eds.), Foundations of Information and Knowledge Systems, First International Symposium, FoIKS 2000, Burg, Germany, February 14-17, 2000 Proceedings, Lecture Notes in Computer Science. Vol. 1762 (2000), Springer. pp. 157–175.

[33] A. Levy, A hierarchy of formulas in set theory. Mem. Amer. Math. Soc. No. 57 (1965), 76 pp. MR 32 N 7399.

[34] A.P. Lisitsa and V.Yu. Sazonov, Delta-languages for sets and LOGSPACE computable graph transformers. Theoretical Computer Science **175**, 1 (1997), 183–222.

[35] A.P. Lisitsa and V.Yu. Sazonov, Linear ordering on graphs, anti-founded sets and polynomial time computability, Theoretical Computer Science **224**, 1–2 (1999), 173–213. (Cf. also earlier version in Proc. of the 4th International Symp. 'Logical Foundations of Computer Science', Yaroslavl, Russia, 1997, Springer LNCS **1234** (1997), 178–188.)

[36] A.P. Lisitsa and V.Yu. Sazonov, Bounded Hyperset Theory and Web-like Data Bases. In: Georg Gottlob, Alexander Leitsch, Daniele Mundici (Eds.): Computational Logic and Proof Theory, 5th Kurt Gödel Colloquium, KGC'97, Vienna, Austria, August 25–29, 1997, Proceedings. LNCS **1289**, Springer, (1997), 172–185.

[37] A.P. Lisitsa and V.Yu. Sazonov, Bounded Hyperset Theory and Web-Like Databases, a manuscript (preliminary version), 1998.

[38] A.B. Livchak, Languages of polynomial queries. Raschet i optimizacija teplotehnicheskih ob'ektov s pomosh'ju EVM, Sverdlovsk, 1982, p. 41 (in Russian).

[39] A.O. Mendelzon, G.A. Mihaila and T. Milo, Querying the World Wide Web. Int. J. on Digital Libraries 1(1): 54-67 (1997).

[40] M. Otto, Bisimulation-Invariant PTIME and higher-dimensional μ-calculus, Theoretical Computer Science **224** (1999), 237–265. (Cf. also earlier version "Capturing Bisimulation-Invariant PTIME" in Proc. of the 4th International Symp. 'Logical Foundations of Computer Science', Yaroslavl, Russia, 1997, Springer LNCS **1234** (1997), 294–305.

[41] V.Yu. Sazonov, Polynomial computability and recursivity in finite domains. Elektronische Informationsverarbeitung und Kybernetik. **16**, N7 (1980) 319–323.

[42] V.Yu. Sazonov, Bounded set theory and polynomial computability. All Union Conference on Applied Logic, Proceedings, Novosibirsk (1985) 188–191 (In Russian).

[43] V.Yu. Sazonov, The Collection Principle and the Existential Quantifier. Logiko-matematicheskie problemy MOZ., Vychislitel'nye sistemy **107** (1985) 30–39 (In Russian). (English translation in: Amer. Math. Soc. Transl. (2) **Vol. 142** (1989) 1–8).

[44] V.Yu. Sazonov, Bounded set theory, polynomial computability and Δ-programming. Application aspects of mathematical logic. Computing systems **122** (1987) 110–132 (In Russian) Cf. also a short English version of this paper in: LNCS **278** Springer (1987) 391–397.

[45] V.Yu. Sazonov, Hereditarily-finite sets with attributes, data bases and polynomial-time computability, Actes Preliminaires du Symposium Franco-Sovietique, INFORMATIKA'91, 16–18 Octobre, Grenoble, INRIA, 1991, 113–132.

[46] V.Yu. Sazonov, Hereditarily-finite sets, data bases and polynomial-time computability. Theoretical Computer Science **119** Elsevier (1993), 187–214.

[47] V.Yu. Sazonov and A.V. Leontjev, On coding hereditarily-finite sets and polynomial-time computability (in Russian), Theoretical and applied foundations of program systems, Program Systems Institute of Russian Acad. of Sci., Pereslavl-Zalessky, 1994, 203–229.

[48] V.Yu. Sazonov, A bounded set theory with anti-foundation axiom and inductive definability. Computer Science Logic, 8th Workshop, CSL'94 Kazimierz, Poland, September 1994, Selected Papers. LNCS **933** Springer (1995), 527–541.

[49] V.Yu. Sazonov, On Bounded Set Theory. Invited talk on the 10th International Congress on Logic, Methodology and Philosophy of Sciences, Florence, August 1995, in Volume I: Logic and Scientific Method, Kluwer Academic Publishers, 1997, 85–103.

[50] V.Yu. Sazonov, Using Agents for Concurrent Querying of Web-like Databases via a Hyper-Set-Theoretic Approach (Extended Abstract), Accepted as a talk on Andrei Ershov Fourth Internatioanl Conference "Perspectives of Systems Informatics", 3–6 July 2001, Novosobirsk, Akademgorodok, Russia, 9 pp.

[51] J.R. Shoenfield, Mathematical Logic. Addison-Wesley, Reading, MA, 1967.

[52] R. Szelepcsényi, 1987, The method of forcing for nondeterministic automata. Bull. European Association Theor. Comp. Sci. (Oct. 1987), 96–100.

[53] M.Y. Vardi, The complexity of relational query languages. Proc. of the 14th. ACM Symp. on Theory of Computing, (1982), 137–146.