A first-principles algebraic approach to data transformations in data cleaning: understanding provenance from the ground up

Santiago Núñez-Corrales
iSchool and NCSA, UIUC

Lan Li
iSchool, UIUC

Bertram Ludäscher
iSchool and NCSA, UIUC

Abstract

We provide a model describing data transformation workflows on tables constructed from first principles, namely by defining datasets as structures with functions and sets for which certain morphisms correspond to data transformations. We define rigid and deep data transformations depending on whether the geometry of the dataset is preserved or not. Finally, we add a model of concurrency using meet and join operations. Our work suggests that algebraic structures and homotopy type theory provide a more general context than other formalisms to reason about data cleaning, data transformations and their provenance.

1 Introduction

Data cleaning [16] is increasingly significant task in data science across industry and academia. Scientific reproducibility depends on the existence of well-curated datasets which implies, in research contexts, respecting specific data storage format specifications, no lexicographic errors or misclassified entities, and adequate representation of missing data in contraposition with valid data. Ensuring all the above can be hampered by manual or systematic errors that may be hard to spot. Datasets are routinely constructed manually via entry interfaces, direct input in spreadsheets or CSV files, with the probability of errors proportional to dataset size and their distribution somewhat randomized. In other cases, automated processes may produce systematic errors in data processing may also be difficult to identify and correct manually. As data volumes and technology grows, the challenge intensifies [6].

Tools such as OpenRefine [10] provide a pragmatic pathway toward better datasets via data cleaning workflows. OpenRefine records the *operation history* applied to a dataset using JSON, also known as a data cleaning *recipe*. Recipes specify data cleaning workflow, sequences of data transformations intended to solve various types of errors in datasets regardless of their origin. The premise of our work is that, to fully reap the benefits of data cleaning, one must first robustly understanding data transformations at depth. The study of data transformations has followed several theoretical paths: using programming language theory [3, 19], relational algebra [4, 2] and general algebraic models [13, 8, 11] or combinations of those [9].

While significant progress has been achieved across these fronts, we believe more progress can still be made with simpler theoretical tools. Simple, in our case, means small syntactic overhead. Acquiring proficiency in many of the existing formalisms demands significant effort just to model simple aspects of data transformations. Provenance-wise, these formalisms tend to focus on the history of a dataset as an atomic unit; we take another route that attempts to also uncover the history of individual cell contents. In the resulting dataset after executing an operation history, each piece of content must be somehow causally connected to those in the original dataset. Parts the original dataset may be not end up in the final dataset, but provenance information should allow us to reconstruct them. We would like to ask specific questions about how a given value in a cell came to be, and what its individual operation history was. Metaphorically, we wish to proceed as when following the trace left by a collection of leaves along a flowing river: many of the leaves will be lost, but for those that survive we should be able to tell their history as long as we know how water carries them. Our aim is to provide a model that promotes cells as first-class citizens with an interesting life of their own.

For this purpose, we harness the language of algebraic structures [15]. We observe that OpenRefine recipes containing data transformation operations are equivalent to a directed lattice of intermediate datasets $\mathscr{D} = \{D_i\}_{i \in I}$ where D_0 is the *original* dataset and D_F the final one connected by functions where the direction of the arrow

is defined by D_i being a predecessor of D_{i+1}, or formally $D_i \prec D_{i+1}$ iff $D_i \xrightarrow{\Delta_j} D_{i+1}$ (equivalently, if $D_{i+1} = \Delta_j(D_i)$) for some transformation Δ_j. Moreover, since the lattice defined above defines a preorder $(\mathscr{D}, \prec)$ in which meet and join operations correspond to concurrent disjoint transformations on copies of a dataset and merge transformations respectively, we can always find a monotone map to a linear preorder of sequential execution steps. All the latter suggests that morphisms constitute an appropriate general language to express facts about data transformations.

Our model resembles the spreadsheet algebra reported in [13] insofar datasets are equated with tables for purposes of simplicity. However, we dispense with the syntactic and operational complexities of relational algebra altogether. The notions developed in this article can be extended to more complex cases where relational algebra operations such as joins and selects occur. While we do not provide extensive discussion on the provenance model arising from our model, the appendix hints at which information can be preserved for such purposes.

2 Defining a Dataset

Intuitively, a dataset is a collection of elements with some algebraic structure. Each element contained in the dataset is a representation of an interesting entity whose representation is a string of symbols drawn from an alphabet Σ. Along with the alphabet, two special strings are provided: the empty string ε for events where the contents may be empty and yet semantically adequate, and the null string $\perp$ for cases when information is legitimately missing. We are now in position to define a language $\mathscr{L}_\Sigma = \Sigma^* \cup \{\perp\}$, which for the time will be taken as regular; hence, to every regular expression $r \in R$ (R being a finite set of regular expressions) corresponds a language $\mathscr{L}_\Sigma(r) \subseteq \mathscr{L}_\Sigma$. From usual data cleaning practices, the contents of individual rows are not free-form, since they depend on certain formatting practices. Assuming that for each of the N columns of a dataset a regular expressions $r_j, 1 \leq j \leq N$ exists, rows are elements of the set

$$\mathscr{R} \subseteq \bigtimes_{j=1}^{N} \mathscr{L}_\Sigma(r_j),$$

which is loosely equivalent to a relation in database theory. But, as indicated, we seek a more flexible representation that decouples the contents from their structural scaffold. We thus define the dataset content as a set of pairs $\mathscr{C} = \{\langle c, \lambda \rangle | c \in C, \lambda \in \mathbb{N}\}$ where

$$C \subseteq \bigcup_{j=1}^{N} \mathscr{L}_\Sigma(r_j), \tag{1}$$

and λ is uniquely assigned for every c before transformations are applied. A dataset is composed of a finite number of rows composed of *cells*, addressable entities by means of row and column indices I, J such that $I = \{1, 2, \cdots, M\}$ and $J = \{1, 2, \cdots, N\}$.

In the usual spreadsheet view, cells are addressable memory locations that contain representations; as expected, we invert this perspective by constructing a function S that assigns a unique index (i, j) elements of $c \in \mathscr{C}$ that appear in the dataset. We call any such function a *structuring* function

$$S : \mathscr{C} \to I \times J \tag{2}$$

$$S = \{c \mapsto (i, j) | c \in \mathscr{C}, i \in I, j \in J\}. \tag{3}$$

We can use the positions of cell contents to compute λ. A simple way to guarantee uniqueness is the mapping $\lambda(i, j) = 2^i \cdot 3^j$. A cell is therefore an object for which the structuring function S *circumstantially* assigns an index, not the other way around. The cell is the mapping $c \mapsto (i, j)$ itself, and S can change as needed. We finally define a *dataset* as the quintuplet

$$D = (R, \mathscr{C}, S, I, J). \tag{4}$$

Several equivalence relations become apparent between any two datasets. D, D' are encoding-equivalent ($D \equiv_R D'$) if they share the same regular expressions. A weaker form of this equivalence occurs by content-preserving transformations between regular expression. They are content-similar ($D \equiv_C D'$) if their content is identical. Both datasets are intention-similar ($D \equiv_S D'$) if a one-to-one isomorphism exists between elements of S in D and those of S' in D'. A curious fact can derived in the following manner. Let us define *dataset equivalence* as the relation

$$D \equiv D' \Leftrightarrow (D \equiv_R D') \wedge (D \equiv_C D') \wedge (D \equiv_S D') \tag{5}$$

and note that such type of equivalence also defines equality, since two datasets sharing the encoding, the structure and the content must be identical. We are therefore justified to state

$$(D = D') \equiv (D \equiv D'). \tag{6}$$

The latter is suggestive of the form of the *univalence axiom* [18]. More work is required to model datasets as types and transformations as homotopies using homotopy type theory (HoTT) [1]. In the same line, we also

suspect that datasets may be equivalent to simplicial sets [14]. Finding such link between data transformations and HoTT would provide a unifying view of data transformations (and by extension data cleaning) potentially capable of, for instance, subsuming relational algebras [7] and providing formal verifiability [5] towards more fundamentally understanding prospective and retrospective provenance.

3 A Data Transformation Algebra

Back to data cleaning pragmatics, we now focus our interest on representing transformations as functions in the space of datasets. For fixed R, $\mathscr{C}$, S, I and J consider the *dataset space*

$$\mathscr{D}_{[R,\mathscr{C},S,I,J]} = \{(R,\mathscr{C},S,I,J) | S \in \mathscr{S}\} \tag{7}$$

over a set of functions $\mathscr{S} \subseteq (I \times J)^{\mathscr{C}}$. For conciseness, we will refer only to those sub-indices of $\mathscr{D}$ relevant to the context in which a transformation occurs. A data transformation is a function

$$\Delta_{\alpha}^{\beta} : \mathscr{D}_{\alpha} \to \mathscr{D}_{\beta} \tag{8}$$

where parametrizations α and β may differ such that for $D \in \mathscr{D}_{\alpha}$ and $D' \in \mathscr{D}_{\beta}$, it holds that $\Delta_{\alpha}^{\beta}(D) = D'$. Depending on which parameters define the space of transformations, we classify them as *geometric transformations* ($\Delta_{[\mathscr{C},S,I,J]}^{[\mathscr{C}',S',I',J']}$), and *rigid transformations* ($\Delta_{\rho}^{\rho'}$) where $\rho \neq [\mathscr{C},S,I,J]$ (i.e. *encoding transformations* ($\rho = [R]$), *content transformations* ($\rho = [\mathscr{C}]$), *structure transformations* ($\rho = [S]$)). We call the pair $s = \langle \alpha, \beta \rangle$ the *signature* of a data transformation, and assume the existence of a function σ that extracts s from Δ.

For any given $\mathscr{D}_{\rho}$, the identity transformation Δ_{id} is always defined such that $\Delta_{\text{id}}(D) = D$. From this, we observe that separate identities exist per dataset space, and, by extension, that data transformations in general either take a dataset space onto itself or onto another space. For instance, changing the encoding or the content within a dataset without altering its shape are all rigid transformations. On the other hand, adding, dropping or removing rows or columns are geometric transformations. Given an operation history $H = \langle \Delta_1, \Delta_2, \cdots, \Delta_n \rangle$, we can compute the corresponding signature set

$$s_H = \{s = \sigma(\Delta_i) | \Delta_i \in H\} \tag{9}$$

and furthermore the signature graph G_H that tracks the motion across signatures induced by the transformation sequence. Consider for instance a dataset transformation that changes the order of rows (Δ_1), replaces one date format by another (Δ_2), fixes certain inconsistent dates (Δ_3) and finally performs a clustering operation to group similar data per those dates (Δ_4). Figure 1 depicts s_H. It becomes immediately apparent that both H and s_H constitute two separate aspects of provenance: H is a prospective account of exactly what happens to the dataset in question, while s_H is a generalized prospective account of how transformations reconfigure the space of transitions between dataset spaces regardless of their specifics.

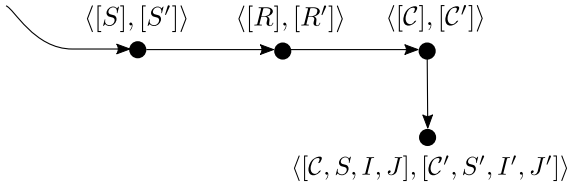

Figure 1: Signature graph s_H for the operation history $H = \langle \Delta_1, \Delta_2, \Delta_3, \Delta_4 \rangle$.

The objects of our algebra are data transformations within an operation history H, and our only operation is composition ($\otimes = \circ$). The application of H to a dataset D_0 should yield a final dataset D_F. Consequently,

$$H(D_0) = \langle \Delta_1, \Delta_2, \cdots, \Delta_n \rangle (D_0) = \left[\bigotimes_{k=n}^{1} \Delta_k \right] (D_0) = \Delta_H(D_0) = D_F \tag{10}$$

Note that composition runs inverse to the order in which data transformations are specified in H, and if no problems exist, defines a transformation itself. The following axioms, along with the previous definition, constitute the basis to determine well-formedness of H for indices k, k', k'':

$$D = \Delta_{\text{id}}(D), \tag{11}$$

$$(\Delta_{\text{id}})^n = \Delta_{\text{id}}, \tag{12}$$

$$\Delta_{\text{id}}^k \otimes \Delta_{\text{id}}^{k'} = \Delta_{\text{id}}^{k''} \Leftrightarrow \sigma(\Delta_{\text{id}}^k) = \sigma(\Delta_{\text{id}}^{k'}) = \sigma(\Delta_{\text{id}}^{k''}), \tag{13}$$

$$\Delta_{\text{id}}^k \otimes \Delta_x^{k'} = \Delta_x \Leftrightarrow \sigma(\Delta_{\text{id}}^k) = \sigma(\Delta_x^{k'}), \tag{14}$$

$$\sigma\left(\left[\Delta_{\alpha}^{\beta}\right]^k \otimes \left[\Delta_{\gamma}^{\delta}\right]^{k'} \right) = \sigma\left(\left[\Delta_{\gamma}^{\delta}\right]^k \otimes \left[\Delta_{\alpha}^{\beta}\right]^{k'} \right) = \langle \alpha \cup \gamma, \beta \cup \delta \rangle. \tag{15}$$

The first axiom captures the absorption-like quality of the identity. The second one describes idempotence. The third axiom ensures that identities are restrained only within their dataset spaces. The fourth one extends the identity restrictions to other data transformations that share the same signature, thus whose outcomes belong to the same space. Finally, the fifth axiom indicates that the composition of different transformations in general

results in the aggregation of parameter spaces. One useful application of the last axiom consists of detecting inconsistencies between the formal specification of transformations: if the signature of a particular composition step is not cumulative, an error has occurred (i.e. transformations refer to unrelatable dataset spaces).

3.1 Rigid Transformations

We now turn our attention to making visible the internal structure of data transformations. At the start, we observe that a clear separation exists between rigid and geometric transformation depending on whether the structure and indexing of the dataset are preserved or not. Let us start by defining a *rigid cell transformation* $\tau(c \mapsto (i,j))$ as the function that, contingent on a predicate p applies a function φ to $c \in \mathscr{C}$ and a function π to pairs (i,j), namely

$$\tau(c \mapsto (i,j)) = \begin{cases} \varphi(c) \mapsto \pi(i,j), p(c) \\ c \mapsto (i,j), \qquad \text{o.w.} \end{cases} \tag{16}$$

We abuse notation here by also assuming that τ correspondingly alters $\mathscr{C}$ and S in the resulting dataset. Note that φ may include translating between formats as captured by regular expressions underlying the languages behind $\mathscr{C}$, choosing new elements form C or even resulting in $\perp$. From the perspective of a cell, τ is reversible if and only if φ is reversible and $p(c') = \neg p(c)$. As an example, adding the number 1 to a numeric cell is reversible ($\varphi^{-1}(c) = c-1$ but the `toUppercase` function is not, since it is surjective (e.g. "AbC" and "aBc" both map to "ABC", and hence $\texttt{toUppercase}^{-1}$ is not a function). In terms of retrospective provenance, reversible functions contain all the information required to reconstruct the past, while irreversible ones require additional records of past values. An interesting example that preserves the geometry of S is that where π corresponds to a permutation of the indices over all c's: since permutations can be labeled and have inverses, they are also reversible.

In order to construct a data transformation, one more feature is required. Up to this point, we have assumed that Δ applies to the entirety of the dataset. In practice, however, few transformations act in this manner. Correcting an individual error is restricted to a single cell, converting from lower to upper case can be a column operation, and rows can be re-indexed. A restricted data transformation $\Delta|_{\{i\}}^{\{j\}}$ with a cell transformation τ is defined as the image of τ over some specific s S applicable to elements where the range are pairs $\{c \mapsto (i,j) | c \in \mathscr{C}\}$, or

$$\Delta|_{\{i\}}^{\{j\}} = \tau[S] \cap \{c \mapsto (i,j) | c \in \mathscr{C}\} \tag{17}$$

Since the restriction indices are sets –i.e. $\{i\} = A$ and $\{j\} = B$- we naturally establish the notation $\Delta|_{*}^{\{j\}} = \Delta|_{I}^{\{j\}}$ for data transformations that operate only on a given column, $\Delta|_{\{i\}}^{*} = \Delta|_{\{i\}}^{J}$ for those that operate on rows and, consequently, $\Delta|_{*}^{*} = \Delta|_{I}^{J} = \Delta$ for data transformations not restricted by structure bounds. More elaborate index sets yield are clearly possible, but avoided here for the sake of clarity since we wish to describe operations that have a single intent per transformation. Looking back to dataset cleaning provenance, data transformation restrictions appear to define essential metadata for both prospective and retrospective analysis. Using index sets, we call two (restricted) dataset transformations $\Delta \equiv \Delta|_{A}^{B}$, $\Delta' \equiv \Delta|_{C}^{D}$ concurrent ($\Delta \| \Delta'$) if $(A \cup C) \times (B \cup D) = \emptyset$ or $(A \cup C) \times (B \cup D) = I \times J$. A curious consequence of this definition is that Δ_{id} is the only dataset transformation covering $I \times J$ in which $\Delta_{\text{id}} \| \Delta$ holds given any other transformation Δ respecting Eqs. 12 and 13. Using concurrency, we further provide commutativity as the axiom

$$\Delta \| \Delta' \Leftrightarrow (\Delta \otimes \Delta') = (\Delta' \otimes \Delta), \tag{18}$$

a necessary step toward, for instance reorganizing data transformation workflows to optimize computational efficiency even when these are sequentially executed.

3.2 Geometric Transformations

Let us now turn our attention to data transformations that change the structure of datasets, not only their content. We start by observing that our definition of rigid cell transformation requires further restrictions. Consider the function $\pi(i,j) = (j,i)$, which corresponds to the transpose of the set. We immediately find that the structure produced by this specification is $S' : \mathscr{C} \to J \times I$. The transformation has changed R, I and J in a fundamentally different manner than that of a permutation. For example, R' contains only one column type after a transposition, namely $\bigcup_j r_j$ that applies to all columns. Similarly, re-utilizing the notation of restricted transformations, we note that both row permutations $\pi(*,j)$ and column permutations $\pi(i,*)$ preserve the geometry of S.

We also find additive or subtractive operations in tools such as OpenRefine. An *additive data transformation* endows the dataset with new rows or columns by means of (a) systematic copying and re-indexing. Conversely, a *subtractive data transformation* removes rows and/or columns systematically. Let us conveniently define $c_{i,j} \equiv c \mapsto (i,j)$, the column addition operation $\Delta^{+}(j', \eta, Y, p)$ transforms the dataset by modifying

$$J' = J \cup |J| + 1 \tag{19}$$

$$S' = \{c_{i,j} | j \leq j'\} \cup \{\eta(Y,i)_{i,j'+1} | j = j'\} \cup \{c_{i,j+2} | j > j'\} \tag{20}$$

.

if the first-order predicate p holds. Note that η is a function that depends on an index $Y \subset J$. For instance, a copy operation is captured by

$$Y = \{j'\} \tag{21}$$

$$\eta(Y,i) = c_{i,\ell}, \ell \in Y \tag{22}$$

while computing an average with a given number of columns and inserting them at a new position can be achieved by

$$Y = \{j'_1, j'_2, ..., j'_k\} \tag{23}$$

$$\eta(Y,i) = \frac{1}{|Y|} \sum_{\ell \in Y}^{j'_k} c_{i,\ell}. \tag{24}$$

Assigning a unique value for λ can be performed by computing $\lambda(i, |J'|)$. The same argument can easily be made for the row addition operation by restating these equations for i' instead of j', and the form of $p(Y)$ captures a wide range of dependencies. In OpenRefine, any operation such a clustering or any other for which a script is expressed solely in terms of η, Y, p is captured by a structural transformation. Note that copies can be traced by looking for columns j_a, j_b with the same values row-wise for which $\lambda(i, j_a)/\lambda(i, j_b) = 3^w$ for $w \geq 1$.

Similarly, let us define a *subtractive column operation* $\Delta^-(j', Y, p)$ with effects

$$J' = J \cup |J| - 1 \tag{25}$$

$$S' = \{c_{i,j} | j < j'\} \cup \{c_{i,j-1} | j > j'\} \tag{26}$$

controlled by the conditional $p(Y)$ as well. Subtractive operations do not alter preexising values of λ. We note that both operations, regardless of their concurrency, do not commute in general ($\Delta^- \otimes \Delta^+ \neq \Delta^+ \otimes \Delta^-$). However, for a given Δ^+ there exists always a Δ^- such that

$$\Delta^- \otimes \Delta^+ = \Delta_{\text{id}}, \tag{27}$$

that is, Δ^- is a left-inverse of Δ^+; this is so since $(\Delta^-)^{-1}$ is injective column-wise. The provenance of additive operations requires no additional information, while the deleted cells need to be preserved in retrospective provenance. Being a left appears to be connected with the need for additional provenance information; this matter requires further investigation.

3.3 Concurrency

We have only considered so far linear operation histories. In reality, data cleaning can be expensive and any opportunities for concurrency are valuable. Let us consider two additional operators $\bigwedge(D) = \langle D, D \rangle$ and $\bigvee(\langle D, D' \rangle) = D''$. The first one (a *fork*) duplicates the dataset via n subhistories; for simplicity, $n = 2$ here. The second operator is a *join*, which yields a dataset such that $R'' = R \cap R'$ (representing a mutual update), $\mathscr{C}'' = \mathscr{C} \cup \mathscr{C}'$, $S'' = (S \cap S') \cup (S \triangle S')$ and $I'' \times J'' = I' \times J' = I \times J$. Error-wise, finding that $|S''| \neq |S|$ implies data conflicts that can be explained by incorrect factorization of dependencies. If $\Delta \| \Delta'$ the following axiom captures the concurrency intent:

$$(\Delta \otimes \Delta')(D) \equiv \tag{28}$$

$$\bigvee \left[\bigwedge \left[(\Delta \otimes \Delta')(D) \right] \right] \equiv \bigvee \left[\Delta(D), \Delta'(D) \right]. \tag{29}$$

An example workflow using all constructions up to this point is provided in the Appendix.

4 Conclusion

In this article, we described a data transformation algebra for data cleaning and provenance. The framework explained here appears to cover most data cleaning operations present in tools such as OpenRefine, while remaining general enough to study workflows (i.e. operation histories) at various levels of abstraction. More work remains to be done regarding isomorphisms between recipes, since it is often of interest to determine the transformability between possibly equivalent operation histories.

Our framework revealed several connections between datasets, data transformations and algebraic structures. In particular, the appearance of an instance of the univalence axiom cast datasets and their transformations as simplicial sets, rich structures under the light of homotopy type theory. Another line of work corresponds to exploring lossy transformations between regular expressions. Finally, a software package following this algebraic approach is under development.

Acknowledgments

The authors acknowledge support by the Center for Informatics Research in Science & Scholarship at the School of Information Sciences. This work has been partially funded by NSF award 1541450 (Whole Tale) and a ACM/Intel SIGHPC Computational and Data Science Fellowship (2017). We thank reviewers for their thoughtful and extensive comments.

References

[1] AWODEY, S., PELAYO, Á., AND WARREN, M. A. Voevodsky's univalence axiom in homotopy type theory. *Notices of the AMS 60*, 9 (2013), 1164–1167.

[2] BERTOSSI, L., KOLAHI, S., AND LAKSHMANAN, L. V. Data cleaning and query answering with matching dependencies and matching functions. *Theory of Computing Systems 52*, 3 (2013), 441–482.

[3] BUNEMAN, P., DAVIDSON, S. B., HART, K., OVERTON, C., AND WONG, L. A data transformation system for biological data sources. *Database Research Group (CIS)* (1995), 12.

[4] CARREIRA, P. J., GALHARDAS, H., LOPES, A., AND PEREIRA, J. L. Extending relational algebra to express one-to-many data transformations. In *Simpósio Brasileiro de Bancos de Dados* (2005), pp. 145–159.

[5] CHU, S., WEITZ, K., CHEUNG, A., AND SUCIU, D. Hottsql: Proving query rewrites with univalent sql semantics. *ACM SIGPLAN Notices 52*, 6 (2017), 510–524.

[6] CHU, X., ILYAS, I. F., KRISHNAN, S., AND WANG, J. Data cleaning: Overview and emerging challenges. In *Proceedings of the 2016 International Conference on Management of Data* (2016), pp. 2201–2206.

[7] CLEMENTINO, M. M., HOFMANN, D., AND MONTOLI, A. Covering morphisms in categories of relational algebras. *Applied Categorical Structures 22*, 5-6 (2014), 767–788.

[8] DIAS, J., OGASAWARA, E., DE OLIVEIRA, D., PORTO, F., VALDURIEZ, P., AND MATTOSO, M. Algebraic dataflows for big data analysis. In *2013 IEEE International Conference on Big Data* (2013), IEEE, pp. 150–155.

[9] GIANNAKOPOULOU, S. A., KARPATHIOTAKIS, M., GAIDIOZ, B. C. D., AND AILAMAKI, A. Cleanm: an optimizable query language for unified scale-out data cleaning. *Proceedings of the VLDB Endowment 10*, CONF (2017).

[10] HAM, K. Openrefine (version 2.5). http://openrefine. org. free, open-source tool for cleaning and transforming data. *Journal of the Medical Library Association: JMLA 101*, 3 (2013), 233.

[11] KHEDRI, R., CHIANG, F., AND SABRI, K. E. An algebraic approach towards data cleaning. *Procedia Computer Science 21* (2013), 50–59.

[12] KOREL, B., AND SMITH, R. Slicing event traces of large software systems.

[13] LIU, B., AND JAGADISH, H. A spreadsheet algebra for a direct data manipulation query interface. In *2009 IEEE 25th International Conference on Data Engineering* (2009), IEEE, pp. 417–428.

[14] PELAYO, Á., AND WARREN, M. Homotopy type theory and voevodsky's univalent foundations. *Bulletin of the American Mathematical Society 51*, 4 (2014), 597–648.

[15] PRIESTLEY, H. A. Ordered sets and complete lattices. In *Algebraic and coalgebraic methods in the mathematics of program construction*. Springer, 2002, pp. 21–78.

[16] RAHM, E., AND DO, H. H. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull. 23*, 4 (2000), 3–13.

[17] TAN, W. C. Research problems in data provenance. *IEEE Data Eng. Bull. 27*, 4 (2004), 45–52.

[18] VOEVODSKY, V. The equivalence axiom and univalent models of type theory. *Talk at CMU* (2010), 172–187.

[19] WEIS, M., AND MANOLESCU, I. Declarative xml data cleaning with xclean. In *International Conference on Advanced Information Systems Engineering* (2007), Springer, pp. 96–110.

Login email	Identifier	First name	Last name
laura@example.com	2070	Laura	Grey
craig@example.com	4081	Craig	Johnson
mary@example.com	9346	Mary	Jenkins
jamie@example.com	5079	Jamie	Smith

Table 1: Minimum Dataset for Email Onboarding from Staffbase

5 Appendix: an complete example

Consider the following small table provided by Staffbase as examples for user onboarding with email[1]. We chose the Minimum Dataset for Email Onboarding (Table 1). We start by specifying the dataset using our constructions, and to that end, we obtain the regular expressions for each column.

- Login email (r_{email}):

```
^([a-zA-Z0-9_\-\.]+)@((\[[0-9]{1,3}\.
[0-9]{1,3}\.[0-9]{1,3}\.)|
(([a-zA-Z0-9\-]+\.)+))([a-zA-Z]{2,4}|
[0-9]{1,3})(\]?)$
```

- Identifier (r_{ident}):

```
[0-9]{3}
```

- First name (r_{fname}):

```
[A-Z][a-z]*
```

- Last name (r_{lname}):

```
[A-Z][a-z]*
```

Hence, $R_0 = \{r_{\text{email}}, r_{\text{ident}}, r_{\text{fname}}, r_{\text{lname}}\}$. The content of the dataset is thus:

$$\begin{aligned}\mathscr{C}_0 = \{&\langle'\text{laura@example.com}', 6\rangle, \langle'\text{craig@example.com}', 12\rangle,\\ &\langle'\text{mary@example.com}', 24\rangle, \langle'\text{jamie@example.com}', 48\rangle,\\ &\langle 2070, 18\rangle\langle 4081, 36\rangle, (9346, 72\rangle, (5079, 144\rangle,\\ &\langle'\text{Laura}', 54\rangle, \langle'\text{Craig}', 108\rangle, \langle'\text{Mary}', 216\rangle, \langle'\text{Jamie}', 436\rangle,\\ &\langle'\text{Gray}', 162\rangle, \langle'\text{Johnson}', 324\rangle, \langle'\text{Jenkins}', 648\rangle, \langle'\text{Smith}', 1296\rangle\}\end{aligned}$$

[1] See: `https://support.staffbase.com/hc/en-us/articles/360007108391-CSV-File-Examples`.

with $I_0 = \{1,2,3,4\}$ and $J_0 = \{1,2,3,4\}$. For clarity purposes, we will omit λ values is S to concentrate on dataset changes. Using that, we proceed to specify its structure:

$$
\begin{aligned}
S_0 = \{ & \\
& '\text{laura@example.com}' \mapsto (1,1), \\
& '\text{craig@example.com}' \mapsto (2,1), \\
& '\text{mary@example.com}' \mapsto (3,1), \\
& '\text{jamie@example.com}' \mapsto (4,1), \\
& 2070 \mapsto (1,2), \\
& 4081 \mapsto (2,2), \\
& 9346 \mapsto (3,2), \\
& 5079 \mapsto (4,2), \\
& '\text{Laura}' \mapsto (1,3), \\
& '\text{Craig}' \mapsto (2,3), \\
& '\text{Mary}' \mapsto (3,3), \\
& '\text{Jamie}' \mapsto (4,3), \\
& '\text{Gray}' \mapsto (1,4), \\
& '\text{Johnson}' \mapsto (2,4), \\
& '\text{Jenkins}' \mapsto (3,4), \\
& '\text{Smith}' \mapsto (4,4) \\
\}. &
\end{aligned}
$$

Thus we define $D_0 = D_{\text{ob}} = (R_0, \mathscr{C}_0, S_0, I_0, J_0)$.

5.1 A set of equivalent histories

Suppose that user names in a hypothetical system we need to populate are equal to user emails without the '@' symbol and the domain part. Take the functional implementation (using Haskell throughout this example) of a function φ_1 = `uname_from_email` that performs such task

```
uname_from_email x = head (splitOn '@' x)
```

with $\pi_1(i,j) = (i,j)$ and $p_1(c) = \text{true}$, we define our first transformation $\Delta_1 = \Delta|^1_*$. Since it is irreversible, we can preserve the tuples

$$(c \mapsto (i,j), \varphi_1(c) \mapsto \pi_1(i,j))$$

only for cells that experienced change as the minimum required information to fully accomplish retrospective provenance. The set prov^k for transformation D_k is the generalized set of such tuples. After applying Δ_1 to D_0 we obtain a dataset such that

$$
\begin{aligned}
\mathscr{C}_1 - \mathscr{C}_0 = \{ & \\
& \langle '\text{laura}', 6\rangle, \langle '\text{craig}', 12\rangle, \\
& \langle '\text{mary}', 24\rangle, \langle '\text{jamie}', 48\rangle \\
\} &
\end{aligned}
$$

and

$$
\begin{aligned}
S_1 = \{ & \\
& '\text{laura}' \mapsto (1,1), \\
& '\text{craig}' \mapsto (2,1), \\
& '\text{mary}' \mapsto (3,1), \\
& '\text{jamie}' \mapsto (4,1), \\
& 2070 \mapsto (1,2), \\
& 4081 \mapsto (2,2), \\
& 9346 \mapsto (3,2), \\
& 5079 \mapsto (4,2), \\
& '\text{Laura}' \mapsto (1,3), \\
& '\text{Craig}' \mapsto (2,3), \\
& '\text{Mary}' \mapsto (3,3), \\
& '\text{Jamie}' \mapsto (4,3), \\
& '\text{Gray}' \mapsto (1,4), \\
& '\text{Johnson}' \mapsto (2,4), \\
& '\text{Jenkins}' \mapsto (3,4), \\
& '\text{Smith}' \mapsto (4,4) \\
\}. &
\end{aligned}
$$

where

$$
\begin{aligned}
\text{prov}^1 = \{ & \\
& ('\text{laura@example.com}' \mapsto (1,1), '\text{laura}' \mapsto (1,1)) \\
& ('\text{craig@example.com}' \mapsto (2,1), '\text{craig}' \mapsto (2,1)) \\
& ('\text{mary@example.com}' \mapsto (3,1), '\text{mary}' \mapsto (3,1)) \\
& ('\text{jamie@example.com}' \mapsto (4,1), '\text{jamie}' \mapsto (4,1)) \\
\}. &
\end{aligned}
$$

Let us now suppose that user creation in the destination system starts with identifier 0 instead of identifier 1 in the source table. Hence, we need a transformation that subtracts 1 to all identifiers $\Delta_2 = \Delta|^{\{2\}}_*$. This is easily achievable with the function φ_2 = `decr_id`

```
decr_id x = x - 1
```

whose inverse φ_2^{-1} exists and is

```
incr_id x = x - 1
```

and hence $\text{prov}^2 = \emptyset$ since invertible maps require no additional recording effort. Correspondingly,

$$\begin{aligned}\mathscr{C}_2 - \mathscr{C}_1 = \{ & \\ & \langle 2069, 18\rangle, \langle 4080, 36\rangle, \\ & \langle 9345, 72\rangle, \langle 5078, 144\rangle \\ \}&\end{aligned}$$

and

$$\begin{aligned}S_2 = \{ & \\ & '\text{laura}' \mapsto (1,1), \\ & '\text{craig}' \mapsto (2,1), \\ & '\text{mary}' \mapsto (3,1), \\ & '\text{jamie}' \mapsto (4,1), \\ & 2069 \mapsto (1,2), \\ & 4080 \mapsto (2,2), \\ & 9345 \mapsto (3,2), \\ & 5078 \mapsto (4,2), \\ & '\text{Laura}' \mapsto (1,3), \\ & '\text{Craig}' \mapsto (2,3), \\ & '\text{Mary}' \mapsto (3,3), \\ & '\text{Jamie}' \mapsto (4,3), \\ & '\text{Gray}' \mapsto (1,4), \\ & '\text{Johnson}' \mapsto (2,4), \\ & '\text{Jenkins}' \mapsto (3,4), \\ & '\text{Smith}' \mapsto (4,4) \\ \}.&\end{aligned}$$

Now, in our hypothetical destination system all users with identifiers above 5000 are administrators due to internal rules, and we would like our dataset to reflect that. We define $\Delta_3 = \Delta_3^+(j', \eta, Y, p)$ where

$$\begin{aligned} j' &= 5, \\ Y &= 2, \\ p(c) &= \text{true} \\ \eta(Y,i) &= \begin{cases} '\text{admin}', & c_{i,\ell} > 5000, \ell \in Y \\ '\text{user}', & \text{o.w.} \end{cases} \end{aligned}$$

Note that this operation is invertible by setting $\Delta_3^{-1} = \Delta_3^-(j', Y, p)$ with $j' = 5$ and $p(y) = \text{true}$. In this case, no provenance information is required either.

As with the prior cases, we compute

$$\begin{aligned}\mathscr{C}_3 - \mathscr{C}_2 = \{ & \\ & \langle '\mathit{user}', 486\rangle, \langle '\mathit{user}', 972\rangle, \\ & \langle '\mathit{admin}', 1944\rangle, \langle '\mathit{admin}', 3888\rangle \\ \}\}&\end{aligned}$$

$$\mathscr{C}_2 - \mathscr{C}_1 =$$

for which its regular expression is r_{group} is

```
'user' | 'admin'
```

so that $R_3 = R_0 \cup \{r_{\text{group}}\}$as well as the structure

$$\begin{aligned}S_3 = \{ & \\ & '\text{laura}' \mapsto (1,1), \\ & '\text{craig}' \mapsto (2,1), \\ & '\text{mary}' \mapsto (3,1), \\ & '\text{jamie}' \mapsto (4,1), \\ & 2069 \mapsto (1,2), \\ & 4080 \mapsto (2,2), \\ & 9345 \mapsto (3,2), \\ & 5078 \mapsto (4,2), \\ & '\text{Laura}' \mapsto (1,3), \\ & '\text{Craig}' \mapsto (2,3), \\ & '\text{Mary}' \mapsto (3,3), \\ & '\text{Jamie}' \mapsto (4,3), \\ & '\text{Gray}' \mapsto (1,4), \\ & '\text{Johnson}' \mapsto (2,4), \\ & '\text{Jenkins}' \mapsto (3,4), \\ & '\text{Smith}' \mapsto (4,4) \\ & '\text{user}' \mapsto (1,5), \\ & '\text{user}' \mapsto (2,5), \\ & '\text{admin}' \mapsto (3,5), \\ & '\text{admin}' \mapsto (4,5) \\ \}&\end{aligned}$$

with $J_3 = \{1,2,3,4,5\}$

Finally, both first and last names by convention in our system are stored using uppercase characters only for compliance with legacy processes. Take the function φ_4 = `toUpper`

```
toUpper = map (\c ->
                if c >= 'a' && c <= 'z'
                then toEnum (fromEnum c - 32)
                else c)
```

and define $\Delta_4 = \Delta|_*^{\{3,4\}}$ such that

$$\begin{aligned}\text{prov}^4 = \{&\\ &('\text{Laura}' \mapsto (1,3), '\text{LAURA}' \mapsto (1,3))\\ &('\text{Craig}' \mapsto (2,3), '\text{CRAIG}' \mapsto (2,3))\\ &('\text{Mary}' \mapsto (3,3), '\text{MARY}' \mapsto (3,3))\\ &('\text{Jamie}' \mapsto (4,3), '\text{JAMIE}' \mapsto (4,3))\\ &('\text{Gray}' \mapsto (1,4), '\text{GRAY}' \mapsto (1,4))\\ &('\text{Johnson}' \mapsto (2,4), '\text{JOHNSON}' \mapsto (2,4))\\ &('\text{Jenkins}' \mapsto (3,4), '\text{JENKINS}' \mapsto (3,4))\\ &('\text{Smith}' \mapsto (4,4), '\text{SMITH}' \mapsto (4,4))\\ \}.&\end{aligned}$$

since `toUpper` is irreversible. The structure, altering $\mathscr{C}_4$ as in the prior cases, respectively becomes

$$\begin{aligned}S_4 = \{&\\ &'\text{laura}' \mapsto (1,1),\\ &'\text{craig}' \mapsto (2,1),\\ &'\text{mary}' \mapsto (3,1),\\ &'\text{jamie}' \mapsto (4,1),\\ &2069 \mapsto (1,2),\\ &4080 \mapsto (2,2),\\ &9345 \mapsto (3,2),\\ &5078 \mapsto (4,2),\\ &'\text{LAURA}' \mapsto (1,3),\\ &'\text{CRAIG}' \mapsto (2,3),\\ &'\text{MARY}' \mapsto (3,3),\\ &'\text{JAMIE}' \mapsto (4,3),\\ &'\text{GRAY}' \mapsto (1,4),\\ &'\text{JOHNSON}' \mapsto (2,4),\\ &'\text{JENKINS}' \mapsto (3,4),\\ &'\text{SMITH}' \mapsto (4,4)\\ &'\text{user}' \mapsto (1,5),\\ &'\text{user}' \mapsto (2,5),\\ &'\text{admin}' \mapsto (3,5),\\ &'\text{admin}' \mapsto (4,5)\\ \}.&\end{aligned}$$

We observe that $\Delta_2 \not\parallel \Delta_3$ since both transformations act sequentially on column 2. The latter yields the following set of equivalent operation histories:

$$\begin{aligned}H^i(D_0) &= \langle \Delta_1, \Delta_2, \Delta_3, \Delta_4 \rangle (D_0)\\ H^{ii}(D_0) &= \langle \Delta_1, \Delta_2, \Delta_4, \Delta_3 \rangle (D_0)\\ H^{iii}(D_0) &= \langle \Delta_1, \Delta_4, \Delta_2, \Delta_3 \rangle (D_0)\\ H^{iv}(D_0) &= \langle \Delta_2, \Delta_3, \Delta_1, \Delta_4 \rangle (D_0)\\ H^{v}(D_0) &= \langle \Delta_2, \Delta_1, \Delta_3, \Delta_4 \rangle (D_0)\\ H^{vi}(D_0) &= \langle \Delta_2, \Delta_1, \Delta_4, \Delta_3 \rangle (D_0)\\ H^{vii}(D_0) &= \langle \Delta_2, \Delta_3, \Delta_4, \Delta_1 \rangle (D_0)\\ H^{viii}(D_0) &= \langle \Delta_2, \Delta_4, \Delta_3, \Delta_1 \rangle (D_0)\\ H^{ix}(D_0) &= \langle \Delta_2, \Delta_4, \Delta_1, \Delta_3 \rangle (D_0)\\ H^{x}(D_0) &= \langle \Delta_4, \Delta_1, \Delta_2, \Delta_3 \rangle (D_0)\\ H^{xi}(D_0) &= \langle \Delta_4, \Delta_2, \Delta_1, \Delta_3 \rangle (D_0)\\ H^{xii}(D_0) &= \langle \Delta_4, \Delta_2, \Delta_3, \Delta_1 \rangle (D_0).\end{aligned}$$

In practice, as in database management systems, such choice opens ample possibilities for optimizing query histories when these have to be executed using sequential processing only. The reader can easily verify that the union of signatures for these equivalent workflows

$$\bigcup_m s_{H^m}$$

is contained in the graph depicted by Fig. 2, which contains two entry points that preserve the dependency between Δ_2 and Δ_3.

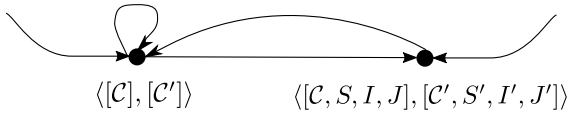

Figure 2: Union of signature graphs s_H for the example operation history over D_{ob}.

5.2 Concurrency

Concurrency-wise, we now choose H^i (any other choice is fine) and derive the following maximally concurrent structure:

$$\begin{aligned}\bigvee \left[\bigwedge [(\Delta_4 \otimes \Delta_3 \otimes \Delta_2 \otimes \Delta_1)(D_0)] \right] &=\\ \bigvee \left[\bigwedge [(\Delta_4 \otimes ((\Delta_3 \otimes \Delta_2) \otimes \Delta_1))(D_0)] \right] &=\\ \bigvee [\Delta_4(D_0), ((\Delta_3 \otimes \Delta_2) \otimes \Delta_1))(D_0)] &=\\ \bigvee \left[\Delta_4(D_0), \bigvee \left[\bigwedge [((\Delta_3 \otimes \Delta_2) \otimes \Delta_1))(D_0)] \right] \right] &=\\ \bigvee \left[\Delta_4(D_0), \bigvee [(\Delta_3 \otimes \Delta_2)(D_0), \Delta_1(D_0)] \right].&\end{aligned}$$

Once again, since $\Delta_2 \not\parallel \Delta_3$, this is the maximally concurrent decomposition that can be achieved. We may improve our notation to make more explicit the concurrency aspect by introducing the following axiom:

$$\begin{aligned}\bigvee\left[\Delta(D),\bigvee\left[\Delta'(D),\Delta''(D)\right]\right] = \\ \bigvee\left[\bigvee\left[\Delta(D),\Delta'(D)\right],\Delta''(D)\right] = \\ \bigvee\left[\Delta(D),\Delta'(D),\Delta''(D)\right]\end{aligned} \tag{30}$$

and, applied to the case in hand, the concurrent decomposition becomes

$$\bigvee\left[\Delta_4(D_0),(\Delta_3\otimes\Delta_2)(D_0),\Delta_1(D_0)\right].$$

5.3 The history of three cells

One of the not so obvious advantages of our representation is the ability to represent explicitly the history of single cells in the final dataset. Since a data transformation is an image of a conditionally applied function over a structure, nothing prevents us from expressing a data transformation as a compositionally-defined morphism, much in the same way as a Kronecker (tensor) product. Let us define the product between one-cell data transformations $\Delta|_{\{a\}}^{\{b\}}\odot\Delta'|_{\{c\}}^{\{d\}}=\Delta''|_{\{a,c\}}^{\{b,d\}}$ such that the respective $\varphi_{a,b},\varphi_{c,d}$ are conditionally applied depending on their indexes and respective predicates. Let us further suppose that any dataset resulting from a given operation history can be further obtained by such means, namely, by constructing a new transformation where each cell is the outcome of a one-cell transformation. Namely, this equates to decomposing an entire operation history transformation in Eq. 11 into a *transformation product* of the form

$$\Delta_H(D_0)=\left(\bigodot_{i\in I,j\in J}\Delta_H|_{\{i\}}^{\{j\}}\right)(D_0). \tag{31}$$

It should be expected that every $\Delta_H|_i^j$ is, actually, a composition of one-cell transformations of length at least that of Δ_H $(=n)$

$$\Delta_H|_{\{i\}}^{\{j\}}=\bigotimes_{k=n}^{1}\Delta_k|_{\{i\}}^{\{j\}}. \tag{32}$$

The same argument regarding the need for additional provenance information when a transformation is irreversible applies. Substituting Eq. 32 into Eq. 31 yields

$$\Delta_H(D_0)=\left[\bigodot_{i\in I,j\in J}\left(\bigotimes_{k=n}^{1}\Delta_k|_{\{i\}}^{\{j\}}\right)\right](D_0). \tag{33}$$

We may think of $\Delta_H|_{\{i\}}^{\{j\}}$ as the single step transformation required to obtain $c_F\mapsto(i_F,j_F)$ in the resulting dataset when departing from D_0. It is worth noting that this history may entail saving complex provenance information, in particular for creation and deletion functions that depend on multiple other cells. However, in principle, there is nothing that prevents us from doing so. We choose in the following exercise to reconstruct the operation history of three cells: (1, 1), (2,2) and (3, 5).

Cell (1,1) is only impacted *locally* by Δ_1, and the effect of the other transformations is equal to that of a one-cell identity. Thus,

$$\begin{aligned}\Delta_1|_{\{1\}}^{\{1\}}=\Delta_1 \\ \Delta_2|_{\{1\}}^{\{1\}},\Delta_3|_{\{1\}}^{\{1\}},\Delta_4|_{\{1\}}^{\{1\}}=\Delta_{\mathrm{id}}|_{\{1\}}^{\{1\}}.\end{aligned}$$

Assuming we use H^i, the one-cell operation history becomes

$$\begin{aligned}\Delta_H|_{\{1\}}^{\{1\}}=\Delta_4|_{\{1\}}^{\{1\}}\otimes\Delta_3|_{\{1\}}^{\{1\}}\otimes\Delta_2|_{\{1\}}^{\{1\}}\otimes\Delta_1|_{\{1\}}^{\{1\}}= \\ \left(\Delta_{\mathrm{id}}|_{\{1\}}^{\{1\}}\right)^3\otimes\Delta_1|_{\{1\}}^{\{1\}}=\Delta_1|_{\{1\}}^{\{1\}}\end{aligned}$$

and correspondingly

$$\mathrm{prov}_H^{(1,1)}=\{(1,'\mathrm{laura@example.com}'\mapsto(1,1),'\mathrm{laura}'\mapsto(1,1))\}$$

where the first number in the triplet is the index of the transformation associated with the change. We now repeat the reasoning above for cell (2,2) and obtain

$$\begin{aligned}\Delta_2|_{\{2\}}^{\{2\}}=\Delta_2 \\ \Delta_1|_{\{2\}}^{\{2\}},\Delta_3|_{\{2\}}^{\{2\}},\Delta_4|_{\{2\}}^{\{2\}}=\Delta_{\mathrm{id}}|_{\{2\}}^{\{2\}},\end{aligned}$$

$$\begin{aligned}\Delta_H|_2^2=\Delta_4|_{\{2\}}^{\{2\}}\otimes\Delta_3|_{\{2\}}^{\{2\}}\otimes\Delta_2|_{\{2\}}^{\{2\}}\otimes\Delta_1|_{\{2\}}^{\{2\}}= \\ \left(\Delta_{\mathrm{id}}|_{\{2\}}^{\{2\}}\right)^3\otimes\Delta_2|_{\{2\}}^{\{2\}}=\Delta_2|_{\{2\}}^{\{2\}},\end{aligned}$$

and

$$\mathrm{prov}_H^{(2,2)}=\emptyset$$

due to reversibility of the operation. Finally, we explore the history of cell (3,5) which does not exist on the original dataset and illustrates both dependencies and the local effect of creation transformations. Note that Δ_3 depends on the outcomes of Δ_2. With this in mind, we observe that

$$\Delta_2|_{\{3\}}^{\{5\}} = \Delta_2$$
$$\Delta_3|_{\{3\}}^{\{5\}} = \Delta_3$$
$$\Delta_1|_{\{3\}}^{\{5\}}, \Delta_4|_{\{3\}}^{\{5\}} = \Delta_{\text{id}}|_{\{3\}}^{\{5\}},$$

and

$$\Delta_H|_3^5 = \Delta_4|_{\{3\}}^{\{5\}} \otimes \Delta_3|_{\{3\}}^{\{5\}} \otimes \Delta_2|_{\{3\}}^{\{5\}} \otimes \Delta_1|_{\{3\}}^{\{5\}} =$$
$$\left(\Delta_{\text{id}}|_{\{3\}}^{\{5\}}\right)^2 \otimes (\Delta_3|_{\{3\}}^{\{5\}} \otimes \Delta_2|_{\{3\}}^{\{5\}}) = \Delta_3|_{\{3\}}^{\{5\}} \otimes \Delta_2|_{\{3\}}^{\{5\}}.$$

The provenance of this set is interesting, since the cell did not exist prior to Δ_3. In essence, we need to provide the most recent point referring to objects in D_2 such that η in Δ_3 can be verified. This leads to the curious provenance set

$$\begin{aligned}\text{prov}_H^{(3,5)} = \{ & \\ & (2, 4081 \mapsto (2,2), 4080 \mapsto (2,2)), \\ & (3, \perp \mapsto (3,5), \text{'user'} \mapsto (3,5)) \\ \}&\end{aligned}$$

where all the information to reconstruct the history of the cell exists independent of the dataset. We provide a GitHub repository that materializes this example using an OpenRefine history[2].

With our approach, it is worth noting that significant efficiency gains may be obtained for data cleaning workflows since most one-cell transformation histories tend to contain few transformations different from the identity. Hence, a data transformation from the perspective of the dataset may often be amenable to compression into transformation products of a small number of compositions of one-cell transformations and not spending computation time on untouched or irrelevant data; our work has resemblance to trace slicing in large software systems [12] and eager provenance tracing in relational data flows [17]. Most significantly, since the construction of the set is cell-wise, more granular opportunities for concurrency exist, being the most extreme case that of data cleaning workflows without inter-column or inter-row dependencies that can be executed using embarrassingly parallel processing only limited to core count.

Furthermore, since the cell history is derivable from the dataset history and vice versa to the best of our current knowledge, our algebra appears to have the ability to sustain two consistent views at different data granularities. The leaves in the river appear indeed to be driven by the greater flow, but have lives of their own.

[2] See: `https://github.com/LanLi2017/parse_orhistory`