

UNIVERSIDADE FEDERAL FLUMINENSE

VICTOR BEZERRA ALENCAR

**Prov-Dominoes: An Exploratory Analysis Approach
for Provenance Data**

NITERÓI

2020

UNIVERSIDADE FEDERAL FLUMINENSE

VICTOR BEZERRA ALENCAR

Prov-Dominoes: An Exploratory Analysis Approach for Provenance Data

Dissertação de Mestrado apresentada
ao Programa de Pós-Graduação em
Computação da Universidade Federal
Fluminense como requisito parcial para
a obtenção do Grau de Mestre em
Computação. Área de concentração:
Engenharia de Sistemas e Informação (ESI).

Orientador:

Leonardo Gresta Paulino Murta

Coorientador:

José Ricardo da Silva Júnior

Coorientadora:

Vanessa Braganholo Murta

NITERÓI

2020

VICTOR BEZERRA ALENCAR

Prov-Dominoes: An Exploratory Analysis Approach for Provenance Data

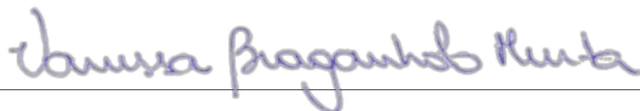
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Engenharia de Sistemas e Informação (ESI).

Aprovada em novembro de 2020.

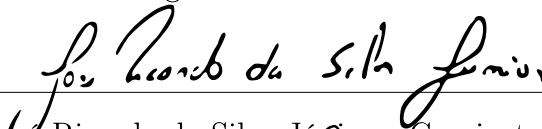
BANCA EXAMINADORA



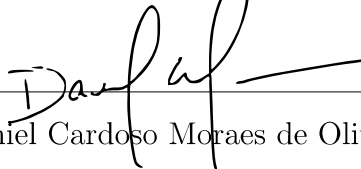
Prof. Leonardo Gresta Paulino Murta – Orientador, UFF



Profa. Vanessa Braganholo Murta – Coorientadora, UFF



Prof. José Ricardo da Silva Júnior – Coorientador, IFRJ



Prof. Daniel Cardoso Moraes de Oliveira – UFF



Profa. Emanuele Marques dos Santos – UFC

Niterói
2020

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

A368p Alencar, Victor Bezerra
Prov-Dominoes : An Exploratory Analysis Approach for
Provenance Data / Victor Bezerra Alencar ; Leonardo Gresta
Paulino Murta, orientador ; José Ricardo da Silva Júnior,
coorientador. Niterói, 2020.
78 f.

Dissertação (mestrado)-Universidade Federal Fluminense,
Niterói, 2020.

DOI: <http://dx.doi.org/10.22409/PGC.2020.m.05183272484>

1. Análise exploratória de dados. 2. Proveniência. 3.
Gpu. 4. Produção intelectual. I. Murta, Leonardo Gresta
Paulino, orientador. II. Silva Júnior, José Ricardo da,
coorientador. III. Universidade Federal Fluminense. Instituto
de Computação. IV. Título.

CDD -

For Landa, my beloved companion.

Acknowledgments

I would like to thank my parents and brother for supporting me along the way.

I am grateful to my cousins, aunts, uncles, and grandparents for providing me wonderful moments.

This dissertation would not have been possible without the help, patience, and counsels of my advisors, Leonardo, José Ricardo, and Vanessa. Also, I would like to thank Troy, for priceless help and counsels.

I want to thank my fellow postgraduate students in the computer science department for promoting a stimulating and welcoming academic and social environment.

I would like to acknowledge the financial, academic and technical support of the Universidade Federal Fluminense, Brazilian Navy, and CASNAV.

Noteworthy, my deepest appreciation goes to Landa, to whom I dedicate this entire dissertation.

And lastly, I thank God for putting all those people and institutions on my way.

Resumo

Proveniência, o registro da história de uma informação, tem se tornado cada vez mais relevante para a compreensão, auditoria e reprodução de tarefas computacionais. Os processos de análise de proveniência muitas vezes podem ser custosos para o usuário devido ao grande volume de dados, aos múltiplos relacionamentos e às informações implícitas em meio a esses dados. Algumas ferramentas existentes fornecem suporte à análise de proveniência com base em diagramas de vínculo entre nós, contando com recursos de visualização sobre arestas e vértices. Outras são ferramentas baseadas em fluxos de trabalho, como VisTrails e Taverna. No entanto, nenhuma delas suporta a exploração de dados de proveniência implícitos, como as inferências das restrições do modelo de dados PROV. Neste trabalho, apresentamos Prov-Dominoes, uma ferramenta projetada para explorar dados de proveniência interativamente. Prov-Dominoes promove as relações de proveniência entre entidades, atividades e agentes em elementos de primeira classe, representados por peças de dominó. Além disso, permite aos usuários combinar tais peças de dominó visual e interativamente, usando GPU. Prov-Dominoes foi avaliado em estudos de caso distintos a fim de observar sua relevância. Foi possível descobrir relações implícitas em um conjunto de dados de características de animais, identificar os parâmetros que influenciaram os resultados da execução de um fluxo de trabalho e destacar as atividades essenciais em uma casa inteligente. Também avaliamos o desempenho de combinações sequenciais executadas em Prov-Dominoes ao lidar com dados de proveniência com milhares de relações, contrastando suas execuções em GPU e CPU. Os resultados mostraram que, para um grande conjunto de dados, GPU superou CPU em duas ordens de magnitude.

Palavras-chave: análise exploratória de dados, proveniência, gpu.

Abstract

Provenance, the record of the history of a piece of information, has become increasingly relevant to understanding, auditing, and reproducing computational tasks. The provenance analysis processes can often be overwhelming to the user due to the large volume of data, the multiple relationships among data, and the implicit information buried into the data. Some existing tools provide provenance analysis support based on node-link diagrams, relying on visualization features over edges and vertices. Others are workflow-based tools, such as VisTrails and Taverna. However, none of them support the exploration of implicit provenance data, such as the inferences of the PROV Data Model Constraints. In this work, we introduce Prov-Dominoes, a tool designed to explore provenance data interactively. Prov-Dominoes promotes the provenance relationships among entities, activities, and agents into first-class elements, represented by domino tiles. Moreover, it allows users to combine and compose such domino tiles visually and interactively, using GPU. We evaluated Prov-Dominoes over distinct case studies, helping us to observe Prov-Dominoes in action. We were able to uncover implicit relationships in a dataset of animal characteristics, identify the parameters that influenced workflow execution results, and highlight essential activities in a smart home. We also evaluated the performance of sequential combinations executed in Prov-Dominoes when dealing with provenance data with thousands of relations, contrasting their executions in GPU and CPU. The results showed that, for a large data set, GPU outperformed CPU by two orders of magnitude.

Keywords: exploratory data analysis, provenance, gpu.

List of Figures

2.1	PROV-DM Types and Relations. Figure taken from Closa et al. [20]. . . .	18
2.2	Provenance graph of beer production according to PROV-DM notation. . .	18
2.3	PROV-N fragment related to parts of Figure 2.2.	19
3.1	Core and Type domino tiles.	23
3.2	PROV-N and matrix enclosing the <i>wasGeneratedBy</i> expressions from the guiding example.	24
3.3	Matrix enclosing <i>entity-type</i> expressions from the guiding example (PROV- N).	25
3.4	Matrix visualization of the <i>wasDerivedFrom</i> domino tile of the guiding example (a), and its centrality graph (b).	26
3.5	Domino tiles combinations through: the left (left-multiplication) (a); or the right (right-multiplication) (b); the top (sum) (c); and the bottom (subtraction) (d).	27
3.6	Transposition of a domino tile.	27
3.7	Domino tile combination: WGB $[E Ac] \times USD [Ac E]$, resulting in a WDF $[E E]$ domino tile.	28
3.8	WGB $[E Ac]$ (a); WGB + WGB (b); and (c) WGB – WGB.	29
3.9	The <i>WDF</i> matrix visualization (a) and the same matrix after the <i>transitive closure</i> operation (b).	31
3.10	$[E T]$ representing some beers and their types (a). $[E T] \times [E T]^T = [E E]$ (b). $[E E]$ after <i>Cluster Sorting</i> (c).	32
4.1	Prov-Dominoes' architecture.	34
4.2	Exploration Provenance Script.	35

4.3	Prov-Dominoes GUI: Command History (a); Domino Tiles List (b); Canvas (c); and Visualization Tabs (d).	36
4.4	WGB combined with USD (in the top), producing WDF (in the bottom): $[E Ac] \times [Ac E] = [E E]$ (a); and WDF matrix content (b).	37
5.1	Head Workflow and its executed modules.	42
5.2	Provenance graph taken from ProvStore.	44
5.3	Classes x Activities ($[Ag T]^T \times [Ac Ag]^T = [T Ac]$).	45
5.4	Ruling Activities x Classes.	46
5.5	EPS of the analysis on entities (a); Resulting domino tile and matrix: outputs in the rows, and inputs and outputs in the columns. The columns represent entities that took part in the output (row) generation (b).	48
5.6	Activities communicating to each other (a); and Activities (rows) associated with agents (b).	51
5.7	<i>Eigenvector centrality</i> graph on activities.	52
5.8	CPU-GPU Comparison with speedups.	56
5.9	CPU-GPU Comparison (Group 1) on the commands time stack.	57
5.10	CPU-GPU Comparison (Group 2) on the commands time stack.	58

List of Tables

2.1	“PROV-DM Relations at a Glance” [41] with trigrams.	21
5.1	Exploratory Practices addressing research questions of the effectiveness evaluation.	54

List of Abbreviations and Acronyms

DAG	<i>Directed Acyclic Graph</i>	12
GPU	<i>Graphical Processing Unit</i>	14
CPU	<i>Central Processing Unit</i>	14
UCI	<i>University of California, Irvine</i>	14
GUI	<i>Graphical User Interface</i>	15
IPAW	<i>International Provenance and Annotation Workshop</i>	17
OPM	<i>Open Provenance Model</i>	17
W3C	<i>World Wide Web Consortium</i>	17
PROV-DM	<i>PROV Data Model</i>	17
PROV-N	<i>PROV Notation</i>	18
EPS	<i>Exploration Provenance Script</i>	34
CUDA	<i>Compute Unified Device Architecture</i>	35
LA4J	<i>Linear Algebra for Java</i>	35
WfMSs	<i>Workflow Management Systems</i>	62
MDS	<i>Multidimensional Scaling</i>	63
PCA	<i>Principal Component Analysis</i>	63
HPC	<i>High-Performance Computing</i>	66

Contents

1	Introduction	12
1.1	Motivation	12
1.2	Goals	13
1.3	Research Methodology	14
1.4	Contributions	15
1.5	Organization	15
2	Background	17
2.1	Provenance and PROV	17
2.1.1	Guiding Example	18
2.1.2	PROV-DM Types	19
2.1.3	PROV-DM Relations	19
2.2	Dominoes, Matrices and GPU	21
3	Prov-Dominoes	22
3.1	Dominoes Game Metaphor	22
3.2	Matrix Representation	23
3.3	Domino Tile Visualizations	25
3.4	Domino Tiles Combinations	26
3.5	Domino Tile Transformations	30
4	Implementation	33
4.1	Prov-Dominoes Architecture	33

4.2	Prov-Dominoes GUI	35
5	Evaluation	39
5.1	Effectiveness	40
5.1.1	Materials and Methods	40
5.1.2	Results and Discussion	43
5.1.3	Exploratory Practices	52
5.2	Efficiency	54
5.2.1	Materials and Methods	54
5.2.2	Results and Discussion	56
5.3	Threats to Validity	58
6	Related Work	60
6.1	Provenance Visualization Analysis	60
6.2	Provenance Data Analysis	62
6.3	Provenance and GPU	64
7	Conclusion	65
7.1	Contributions	65
7.2	Limitations	66
7.3	Future Work	67
	References	69
	Appendix A - PROV-N from the guiding example	73
	Appendix B - List of all EPS commands	74
	Appendix C - Efficiency Evaluation EPS	75

Chapter 1

Introduction

1.1 Motivation

The problem of systematically capturing and managing the provenance¹ for computational tasks has increasingly received attention because of its relevance to a wide range of domains and applications [31], such as bioinformatics, astronomy, and engineering. Without provenance, it becomes difficult to reproduce and share results, solve problems collaboratively, validate results with different input data, and understand the process used to solve a particular problem. Additionally, data products' longevity becomes limited without precise and sufficient information about its generation process, which tends to diminish its value significantly [53].

The provenance of objects (digital or not) can be represented by a *Directed Acyclic Graph* (DAG) of causality, enriched with annotations [44]. This type of graph is useful for understanding the process of entities' generation. In the graph, the edges point to the past, indicating what has been done so far. It is worth noting that the provenance graph differs from a data or control flow graph, where the edges point to what will be executed in the future.

During the provenance analysis, the researcher must make explicit any implicit information that could otherwise be inferred only from context [47]. Although the visualization of provenance in the form of graphs is common, such structures can be limited when there is a need to combine data to uncover implicit information. Besides, provenance graphs can be composed of thousands of vertices and edges, revealing challenges such as the competence to obtain holistic perspectives over large provenance data. Moreover, when performing explorations on large provenance graphs, navigating on this structure becomes

¹Information about entities, activities, and people involved in producing a piece of data or thing [40].

compromised, opening an avenue for more concise ways of visualization analysis.

Most of the existing provenance tools to assist exploratory analysis [38, 28, 3, 26, 39] are based on node-link diagrams, relying on visualization features over edges and vertices for their comprehension. Other tools, such as VisTrails [9] and Taverna [34], provide infrastructure for data exploration and visualization through workflows. Both workflows and node-link diagrams do not support provenance data combinations. For instance, they do not allow the derivation of implicit provenance data, such as constraints [13]. By unveiling implicit data, new analysis possibilities may arise. Besides that, they do not scale well for large provenance data, where the navigation starts to become unfeasible.

1.2 Goals

Given the aforementioned motivation, the aim of this work is to present a new approach for provenance data analysis. We introduce Prov-Dominoes, an approach for interactive explorations of provenance data through the dominoes game’s metaphor. Prov-Dominoes allows fast exploration of implicit information in provenance data. Each existing relationship in the provenance data among entities, activities, and agents is represented by a basic domino tile, which can be visually and interactively combined with other existing tiles to produce derived tiles. For instance, users can combine a basic tile that contains the entities used by activities with another basic tile that contains the entities generated by activities, producing a derived tile that contains the entities derived from other entities.

Under the hood, Prov-Dominoes represents domino tiles as matrices, as they allow fast data combination and concise visualizations. According to Wu et al. [56], “matrix visualization is a graphical technique that can simultaneously explore the associations of up to thousands of subjects, variables, and their interactions, without first reducing dimension”. When analyzing large provenance graphs, matrix visualization allows a more concise visualization of the relationships on the graph, without extensive navigation on a large graph structure.

Summing up our goals on exploring large provenance data:

- Benefit from concise visualizations towards holistic perspectives;
- Interactively combine data to enrich analysis;
- Expose implicit data to ease information extraction; and

- Boost performance by taking advantage of parallel processing such as *Graphical Processing Unit* (GPU).

1.3 Research Methodology

We defined two research questions in this work in order to achieve our goals. The first research question addresses an effective evaluation broken down into three sub-questions. The second research question addresses an efficiency evaluation. Then, our evaluation has as main objective to answer the following research questions:

- *How effective is Prov-Dominoes in supporting exploration of provenance data?*
 - How Prov-Dominoes uncovers implicit information?
 - How Prov-Dominoes provides a holistic perspective?
 - How Prov-Dominoes supports concise analysis?
- *How efficient is Prov-Dominoes when running in GPU in comparison to Central Processing Unit (CPU)?*

We evaluated Prov-Dominoes over four distinct case studies, designed as an attempt to answer the above research questions.

The first case study uses provenance from the *University of California, Irvine* (UCI) [27] Zoo dataset, which contain animal characteristics. This case study aimed at analyzing the capabilities of Prov-Dominoes on extracting implicit and holistic relations from data.

The second case study uses provenance from a VisTrails' workflow sample called "Head." The workflow gathers data from *The Visible Human Project* [1] to render both bones and skin of a head into a volumetric image. This case study aimed to assess the aid of Prov-Dominoes on understanding which activities and parameters were essential to the results obtained by the workflow execution.

The third case study uses provenance captured from a smart home service, where we investigate which activities are central to the service's functioning. Finally, the last case study relies on a large provenance data collected from Twitter to contrast GPU and CPU performances.

In the fourth case study, we collected large provenance data from Twitter to subside a performance assessment of Prov-Dominoes, contrasting GPU and CPU processing modes

of the tool.

1.4 Contributions

This work introduces a novel approach for exploratory analysis of provenance data (Section 3.1). Based on domino tile combinations, the user can visually and interactively explore data. Additionally, we identified a set of agnostic exploratory practices (Section 5.1.3) that serve as an exploration guideline that is independent of the provenance domain.

The concise visualizations (Section 3.3) used in our approach proved assistful to exploratory analysis of provenance data. While unveiling implicit data, new avenues of visualization analysis and combinations arises. The matrix visualization enabled pattern data detection and rapid identification of large data relationships.

The explorations performed in the case studies of the effectiveness evaluation (Section 5.1) showed the potential of Prov-Dominoes for providing a holistic understanding of the relations between agents and activities and unveiling relevant implicit information. Moreover, we show how applying provenance inferences (Section 3.4) can enrich provenances scarce in distinct relations.

Approaching provenance data as matrices (Section 3.2) allowed fast data combinations by taking benefit from GPU. The performance assessment (Section 5.2) over the efficiency case study showed that GPU was 127 times faster in executions involving combinations of thousands of relations.

1.5 Organization

This work is organized in five other chapters, besides this introduction. Chapter 2 provides some background about provenance and GPU. Additionally, Chapter 2 presents a guiding scenario that is explored throughout the following chapters.

Chapter 3 introduces Prov-Dominoes. We detail the dominoes game metaphor and its possibilities. Moreover, we explain how Prov-Dominoes represent provenance concepts, and how they can be visualized, combined, and transformed.

Chapter 4 details the Prov-Dominoes architecture, and its *Graphical User Interface* (GUI). We discuss the three layers of the architecture and how each of its components orchestrate together. Finally, we overview the tool's GUI, highlighting its main parts and

features.

In Chapter 5, we present three distinct case studies to evaluate Prov-Dominoes considering the research questions described in Section 1.3. Besides, we detail the exploratory practices presented as methods to address the research questions and share the results of our explorations.

Chapter 6 presents the related work and highlights how Prov-Dominoes distinguishes itself from other provenance tools, and Chapter 7 concludes this work, presenting its contributions, limitations, and future work.

Chapter 2

Background

2.1 Provenance and PROV

Provenance is well understood in the context of art or digital libraries, where it respectively refers to the documented history of an art object or the documentation of processes in a digital object's life cycle [21]. In 2006, at the *International Provenance and Annotation Workshop* (IPAW) [33], the participants were interested in data provenance, documentation, derivation, and annotation. As a result of the Provenance Challenges [45] presented at IPAW, the *Open Provenance Model* (OPM) [43] was created. In 2013, another provenance model was developed as a *World Wide Web Consortium* (W3C) [5] effort, named PROV [40], which can be viewed as the successor of OPM.

PROV has defined a provenance data model, called *PROV Data Model* (PROV-DM) [4], to support the interoperability and interchange of provenance in heterogeneous environments such as the web. At its core, PROV-DM describes the use and production of entities by activities, which may be influenced in various ways by agents [4]. According to the PROV-DM, entities, activities, and agents are PROV-DM Types. Such types relate to each other through PROV-DM Relations, as shown in Figure 2.1. PROV-DM defines both types and relations as PROV-DM Concepts, and represents provenance using such provenance concepts.

In this chapter, we introduce the fundamental aspects of PROV-DM, considered crucial to this work. The Sections 2.1.2 and 2.1.3, respectively, refer to the types and relations concepts of provenance. This discussion is supported by a guiding example, detailed in Section 2.1.1.

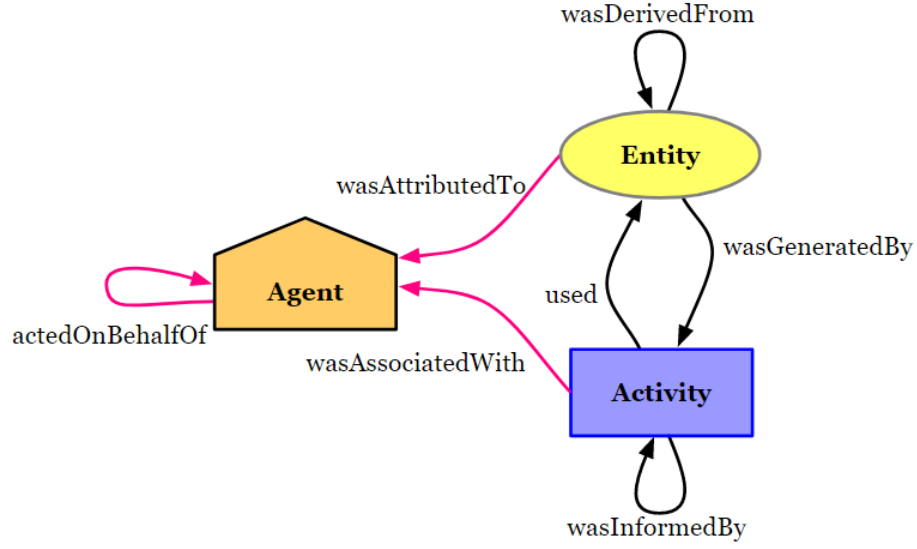


Figure 2.1: PROV-DM Types and Relations. Figure taken from Closa et al. [20].

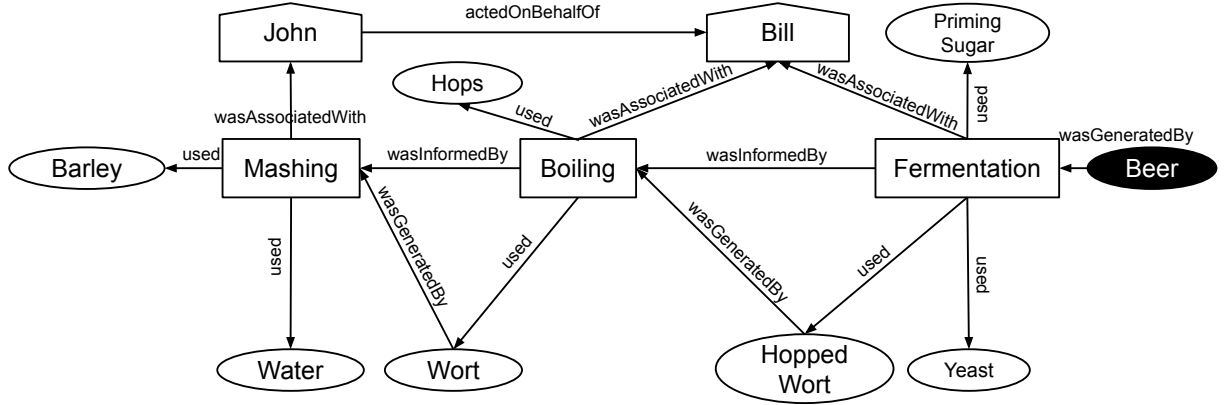


Figure 2.2: Provenance graph of beer production according to PROV-DM notation.

2.1.1 Guiding Example

The concepts that will be used throughout this work consider that *John* (an *agent*) is learning with *Bill* (also an *agent*) how to make *Beer* (an *entity*). In Figure 2.2, we show a provenance graph where *Bill* is responsible for *Boiling* and *Fermentation* (both *activities*) and delegates *Mashing* (another *activity*) to *John*, in order to produce the main *entity*, *Beer* (highlighted in black).

Provenance graphs, such as the one in Figure 2.2, can be textually represented using the *PROV Notation* (PROV-N) [17], a syntax designed to write PROV-DM instances. The notation adopts a functional-style syntax consisting of a predicate name (referring to a PROV-DM Concept) and an ordered list of terms. The predicate and its terms compose a PROV-N expression. The expressions in the PROV-N fragment shown in Figure 2.3 (PROV-N for this example is available in full at Appendix A) represents some

```

...
1 activity(Mashing)
2 activity(Boiling)
3 activity(Fermentation)
4 entity(Barley, [prov:type="Grain"])
5 entity(Wort, [prov:type="Liquid"])
6 entity(Hopped_Wort, [prov:type="Liquid"])
7 entity(Beer, [prov:type="Liquid"])
8 wasGeneratedBy(Wort, Mashing)
9 wasGeneratedBy(Hopped_Wort, Boiling)
10 wasGeneratedBy(Beer, Fermentation)
11 agent(Bill)
...

```

Figure 2.3: PROV-N fragment related to parts of Figure 2.2.

parts of the provenance graph in Figure 2.2. We can see that the expression whose predicate name is a PROV-DM Relation has at least two terms separated by a comma, e.g.: `wasGeneratedBy(Beer, Fermentation)`. The first indicates the origin of the edge in the provenance graph and the second its destiny.

2.1.2 PROV-DM Types

According to PROV-DM, “an *entity* (E) is a physical, digital, conceptual item or anything with some fixed aspects” [4]. An *activity* (Ac) “is something that occurs over a period of time and acts upon or with entities” [4]. An *agent* (Ag) “is something that bears some form of responsibility for an *activity* taking place, for the existence of an *entity*, or for another *agent*’s *activity*” [4]. In the example from Figure 2.2, *Barley* is an *entity*, *Mashing* is an *activity*, and *Bill* is an *agent*.

Additionally, to each *entity*, *activity*, or *agent*, PROV-N provides an expression to define a set of attribute-value pairs, as in line seven of Figure 2.3. One of such attributes, the “*prov:type*”, provides typing capabilities: *Beer* is defined as an *entity* of type “Liquid”.

2.1.3 PROV-DM Relations

According to PROV-DM, there are seven core relations of provenance: generation (*wasGeneratedBy*), usage (*used*), communication (*wasInformedBy*), derivation (*wasDerivedFrom*), association (*wasAssociatedWith*), attribution (*wasAttributedTo*), and delegation (*actedOnBehalfOf*). Two of these relations, *wasGeneratedBy* and *used*, represent explicit interactions between activities and entities. The *wasGeneratedBy* relation indicates the end of the production of an *entity* by an *activity*. Such an *entity* “did not exist before the

generation and becomes available for use after the generation” [4]. The effect of ending production is instantaneous [4]. In the guiding example in Figure 2.2, *Wort wasGeneratedBy Mashing*. The *used* relation indicates the use of an *entity* by an *activity*. The effect of this use is also instantaneous [4]. In our guiding example, *Mashing* used *Water*.

Two other relations, *wasInformedBy* and *wasDerivedFrom*, represent implicit interactions between activities and entities. The *wasInformedBy* relation indicates the exchange of unspecified entities between two activities, a_1 and a_2 , where a_2 uses an *entity* generated by a_1 . Such communication implies that a_2 depends on a_1 , since a_2 requires an *entity* generated by a_1 [4]. In the guiding example from Figure 2.2, *Boiling wasInformedBy Mashing*, since the *Boiling* activity uses the wort generated by the *Mashing* activity. The *wasDerivedFrom* relation is the transformation of an *entity* into another *entity*, an update of an *entity* resulting in another *entity*, or the construction of a new *entity* based on pre-existing entities [4]. In our guiding example, we see that *Mashing* used *Water* and *Wort wasGeneratedBy Mashing*. Although there is no explicit expression indicating that the *Wort* entity was derived from the use of *Water*, by inference [15] we could enrich the graph indicating that *Wort wasDerivedFrom Water*.

Three relations, *wasAssociatedWith*, *wasAttributedTo*, and *actedOnBehalfOf*, represent interactions that involve agents. The *wasAssociatedWith* relation indicates that an *agent* played a role in an *activity* [4]. In the guiding example, *Fermentation wasAssociatedWith Bill*. The *wasAttributedTo* relation indicates that an *entity* was generated by some *activity* associated with a given *agent*. Such a relation is useful when the *activity* is not known or is irrelevant [4]. In our example, we could enrich the graph indicating that *Wort wasAttributedTo John*, because *John* is associated with the *activity* that generated the *Wort*. The *actedOnBehalfOf* relation indicates the assignment of authority and responsibility for an *agent* to carry out a specific *activity* as a delegate or representative of the consigning *agent*. The consigning *agent* has some responsibility for the *activity* carried out by his delegate or representative [4]. In our guiding example, *John actedOnBehalfOf Bill*.

Summarizing, while not all PROV-DM Relations are binary, they all involve two primary elements, referred to as subject and object [4]. Table 2.1 indexes the seven core relations according to their two primary elements [41].

Table 2.1: “PROV-DM Relations at a Glance” [41] with trigrams.

Relation	Subject	Object	Trigram
wasGeneratedBy	Entity	Activity	WGB
used	Activity	Entity	USD
wasDerivedFrom	Entity	Entity	WDF
wasAssociatedWith	Activity	Agent	WAW
wasAttributedTo	Entity	Agent	WAT
wasInformedBy	Activity	Activity	WIB
actedOnBehalfOf	Agent	Agent	AOB

2.2 Dominoes, Matrices and GPU

The domino-matrix abstraction was first introduced by Dominoes [23], a tool designed to assist exploratory analysis on Git repositories. Dominoes uses parallel processing to efficiently process large matrices representing relationships among Git repository data (e.g., developers, commits, modified files). For instance, the combination of two tiles is achieved by multiplying the underlying matrices in GPU. Similarly to the dominoes game, where two tiles can only be combined if they have compatible edges, two matrices can only be multiplied if they have compatible rows/columns. Such efficient GPU processing is possible because matrix multiplication is a highly parallel operation with little reuse of input data [29].

Besides multiplication, the use of matrix as an underlying data structure to represent domino tiles also allows for other operations, such as transitive closure and centrality. These two operations are possible for adjacency matrix, which is a square matrix used to represent a finite graph [32]. Transitive closure can be thought of as establishing a data structure that makes it possible to solve reachability questions (can I get to x from y?) efficiently [54]. Centrality indices are answers to the question “*What characterizes an important vertex?*” The answer is given in terms of a real-valued function on the vertices of a graph, where the values produced are expected to provide a ranking which identifies the most important nodes [6].

The possibility of representing a graph as a matrix acts as a link between data analysis and visualization analysis. Interchanging between the two for different analysis perspectives contributes to enrich the analysis.

Chapter 3

Prov-Dominoes

In this chapter, we present Prov-Dominoes¹, our exploratory analysis tool for provenance data. Prov-Dominoes is compatible with the PROV-N notation, allowing its adoption in different domains and applications. Throughout this section, we detail the dominoes game metaphor along with its features, the Prov-Dominoes' architecture, and its GUI. Sections 3.1 to 3.5 show how Prov-Dominoes represents PROV-DM Concepts and how they can be visualized, combined, and transformed. Section 4.1 presents the technologies and architectural components of the tool. Finally, Section 4.2 provides an overview of the tool's GUI, highlighting its main parts and features.

3.1 Dominoes Game Metaphor

We took the dominoes game as an inspiration to represent the elements in a provenance graph. Examining a provenance graph, we observe that the vertices are PROV-DM Types and the edges PROV-DM Relations. Thus, we took into account the following facts when designing Prov-Dominoes:

- The connections in a provenance graph are directed, associating a subject PROV-DM Type (e.g., *Mashing*) to an object PROV-DM Type (e.g., *Water*); and
- Each connection encloses a PROV-DM Relation (e.g., *Mashing* ***used*** *Water*, in Figure 2.2).

Consequently, each domino tile in Prov-Dominoes represents a PROV-DM Relation with the subject and object of the relation in the left and right edges, respectively. Then,

¹Prov-Dominoes is available at <https://gems-uff.github.io/prov-dominoes>.

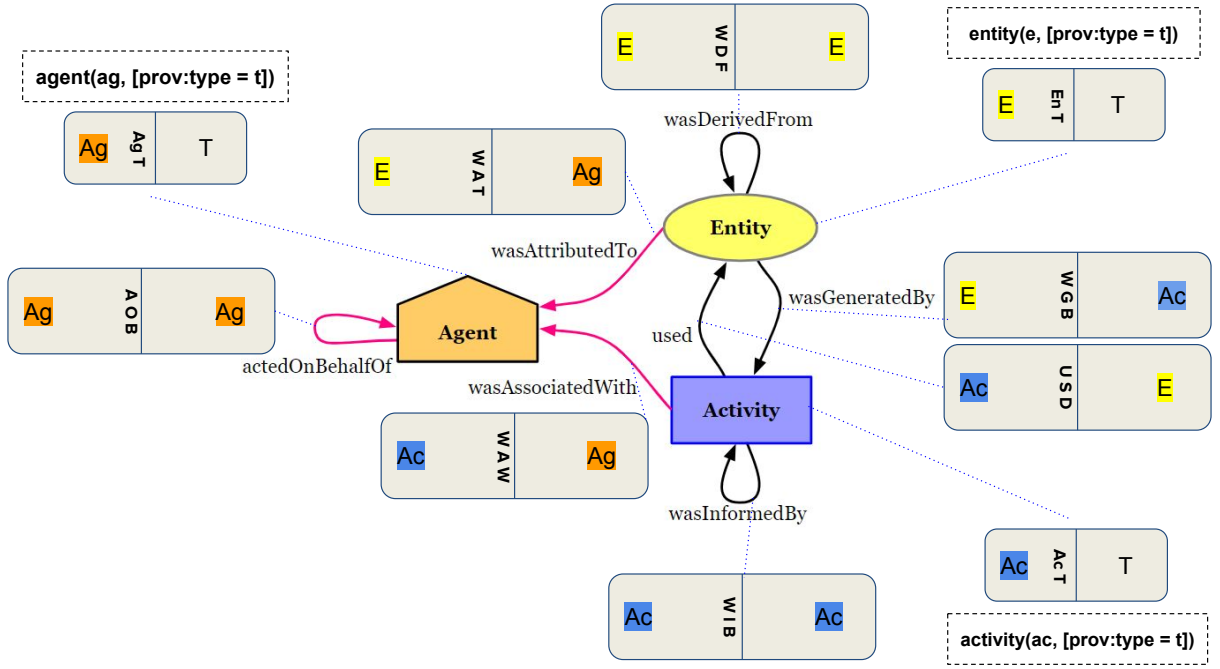


Figure 3.1: Core and Type domino tiles.

we nominate such a domino tile with an uppercase trigram (three capital letters) that indicates which PROV-DM Relation the tile is representing. The trigrams are based on the main letters of the relation names defined by PROV-DM, thus enabling rapid identification of the underlying PROV-DM Relation (e.g., $USE_D \rightarrow USD$, as shown in the “trigram” column of Table 2.1). Finally, we refer to the domino tiles through their trigrams and subject-object edges (e.g., $WGB[E|Ac]$ represents a relation *wasGeneratedBy*, which associates *entities* with *activities*).

The seven core PROV-DM relations were shaped as seven **core domino tiles**. Additionally, we developed three provenance **type domino tiles** to represent types of *agents*, *activities*, and *entities* expressed by the “*prov:type*” attribute, one for each PROV-DM Type: *agent-type* $[Ag|T]$, *activity-type* $[Ac|T]$, and *entity type* $[E|T]$. For instance, the fourth line of Figure 2.3 indicates that the *entity* (E) *Barley* is of type (T) *Grain*. The core and type domino tiles are depicted in Figure 3.1.

3.2 Matrix Representation

Each core domino tile is represented as a matrix associated with a PROV-DM Relation. As each PROV-DM Relation has an expression in PROV-N, whose predicate name is the relation’s name, the matrix can be built based on the PROV-N expressions associated with

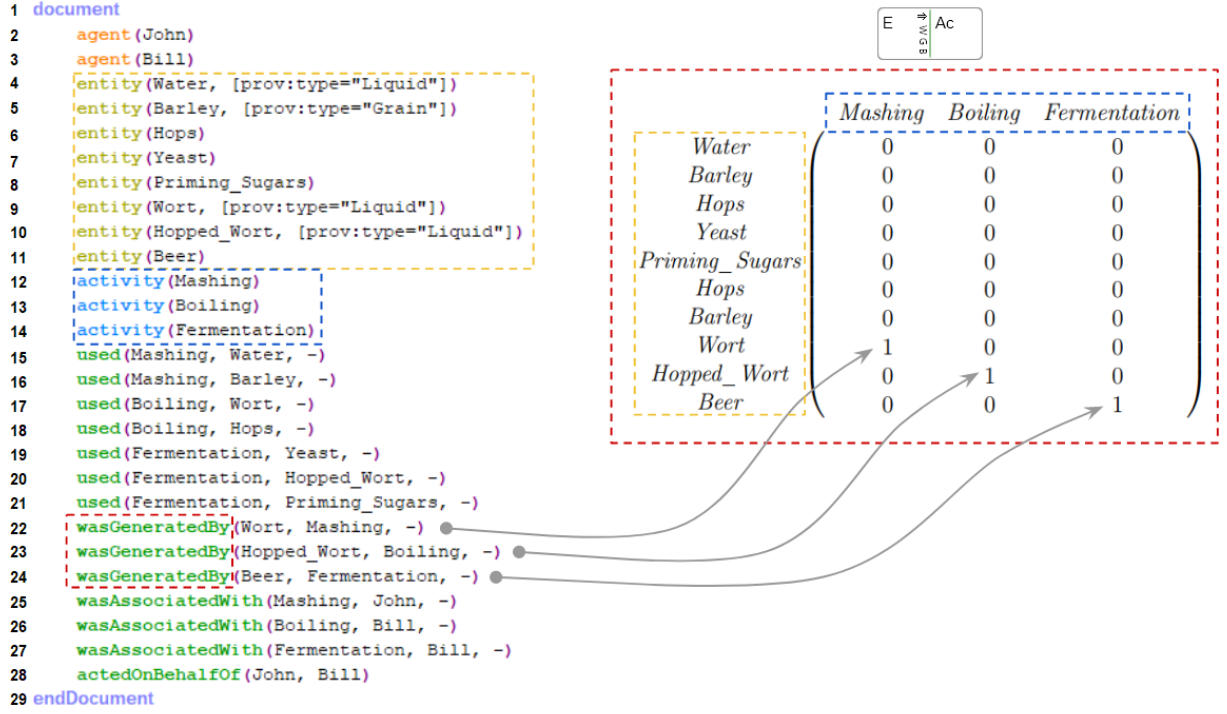


Figure 3.2: PROV-N and matrix enclosing the *wasGeneratedBy* expressions from the guiding example.

the PROV-DM Relation. PROV-N expressions have two required terms for representing relations. Thus, porting to a matrix representation, the first required term composes a row, and the second required term composes a column.

For instance, in Figure 3.2, the expression in the 22th line (*wasGeneratedBy*(*Wort* , *Mashing*)) has the predicate *wasGeneratedBy* (referring to the generation relation) and two required terms (*Wort* and *Mashing*). The first required term (*Wort*) becomes a row index in the WGB matrix, representing an instance of *entity* (E, subject of the generation relation). The second required term (*Mashing*) becomes a column index in the WGB matrix, representing an instance of *activity* (Ac, object of the generation relation). The cell $WGB_{i,j}$ of the matrix is set by counting the number of expressions with the same pair of required terms. In this case: $WGB_{Wort,Mashing} = 1$. The 23th and 24th lines have the same *wasGeneratedBy* predicate, then their pair of terms (*Hopped_Wort*, *Boiling*) and (*Beer*, *Fermentation*) are set: $WGB_{Hopped_Wort,Boiling} = 1$ and $WGB_{Beer,Fermentation} = 1$, respectively. The resulting matrix that encloses the *wasGeneratedBy* expressions from the guiding example (PROV-N) is shown in Figure 3.2 (matrix).

In the case of domino tiles' type, the same idea of pair of indexes to map cells work, where the PROV-DM Type indexes the row, and the value of the defined "prov:type" indexes the column. For instance, in the 4th line (*entity*(*Water* , [prov:type="Liquid"])

	<i>Grain</i>	<i>Liquid</i>
<i>Water</i>	0	1
<i>Barley</i>	1	0
<i>Hops</i>	0	0
<i>Yeast</i>	0	0
<i>Priming_Sugars</i>	0	0
<i>Wort</i>	0	1
<i>Hopped_Wort</i>	0	1
<i>Beer</i>	0	1

Figure 3.3: Matrix enclosing *entity-type* expressions from the guiding example (PROV-N).

of Figure 3.2, the cell $\text{EnT}_{\text{Water}, \text{Liquid}}$ of the matrix EnT enclosing *entity-type* [E|T] is set to one. The resulting matrix that encloses the *entity-type* expressions (lines 4 to 11) in Figure 3.2 is shown in Figure 3.3.

3.3 Domino Tile Visualizations

Prov-Dominoes has two kinds for content visualization of the domino tiles: matrix and centrality graph. They were chosen due to their concise visualization properties. The matrix visualization is a tabular representation of the matrix, where the cells are colored according to their values. Grey indicates zero, blue indicates the maximum value present in the matrix, and white indicates the minimum value different from zero. Values between maximum and minimum are colored in shades of blue (from blue to white).

For instance, consider the underlying matrix of the domino tile *wasDerivedFrom*, whose generation is represented by Figure 3.7 and explained in Section 3.4, shown in Figure 3.4.a. The blue cells illustrate the maximum value (1), indicating that *Wort* was derived from *Water* and *Barley*. Such matrix visualization allows a more concise identification of the relationships on the graph without extensive navigation on a large graph structure.

The centrality graph visualization is an *eigenvector centrality* implementation. Centrality measures indicate that some nodes are more important (central) than others in a graph. The idea of centrality was first introduced in the context of social systems, where the location of an individual in the network may correlate with its influence or power in group processes [22]. The *eigenvector centrality* (also called prestige score [58]) measures such importance through recursive connections. A high eigenvector score means that a

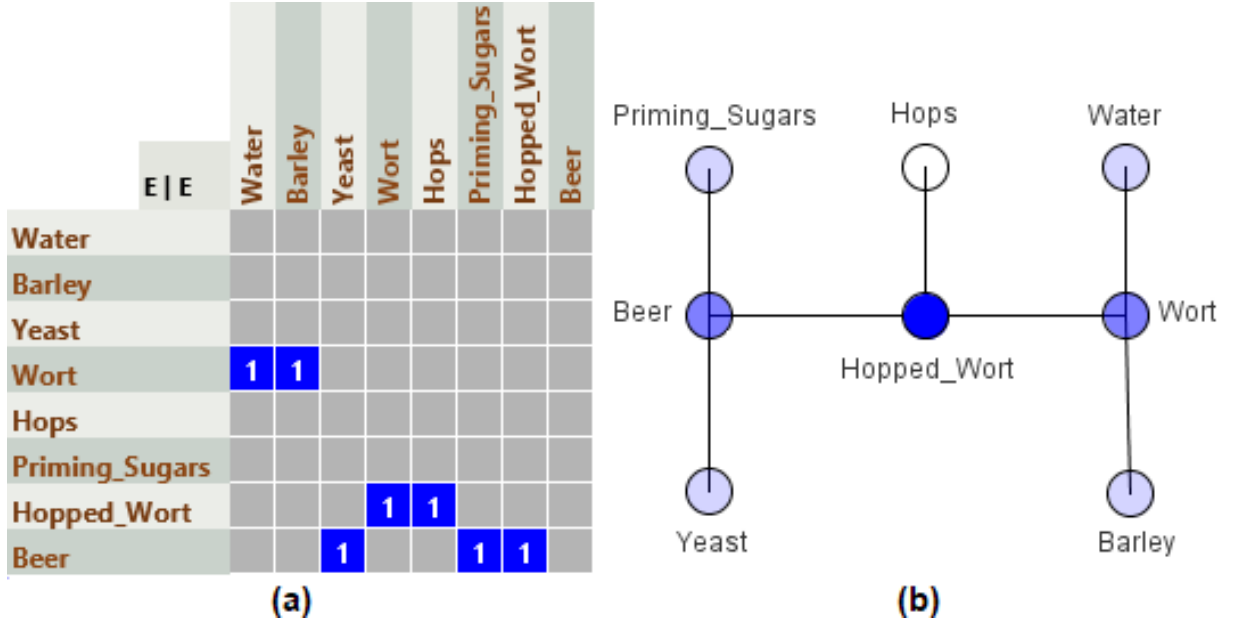


Figure 3.4: Matrix visualization of the *wasDerivedFrom* domino tile of the guiding example (a), and its centrality graph (b).

node is connected to many nodes with high scores [48]. Because this centrality visualization requires a graph, it only makes sense for adjacency matrices encoded in double domino tiles (with the same subject-object edges). When analyzing the square matrix of a double domino tile, it is not clear which cells have more importance over others, as the cell's value only accounts for occurrence. The centrality graph provides scores for an importance perspective.

For example, Figure 3.4.b exhibits the *eigenvector centrality* graph for the *wasDerivedFrom* domino tile of our guiding example. The graph suggests *Hopped_Wort* as the most important entity in beer production. The nodes are colored in shades of blue, where the higher score node is blue and the lower white. Such specific centrality graphs focus on just one PROV-DM Type (e.g., entities, in Figure 3.4.b) and provide a shorter visualization compared to the whole provenance graph. Besides, the nodes' score provide means to navigate such reduced graphs in a more consistent direction, following the high or low centrality scores.

3.4 Domino Tiles Combinations

In Prov-Dominoes, data combination is achieved by merging two domino tiles into one. Figure 3.5 illustrates the three existing types of combinations: by approaching part-equal²

²Only the edges and dimensions being approached should be equal.

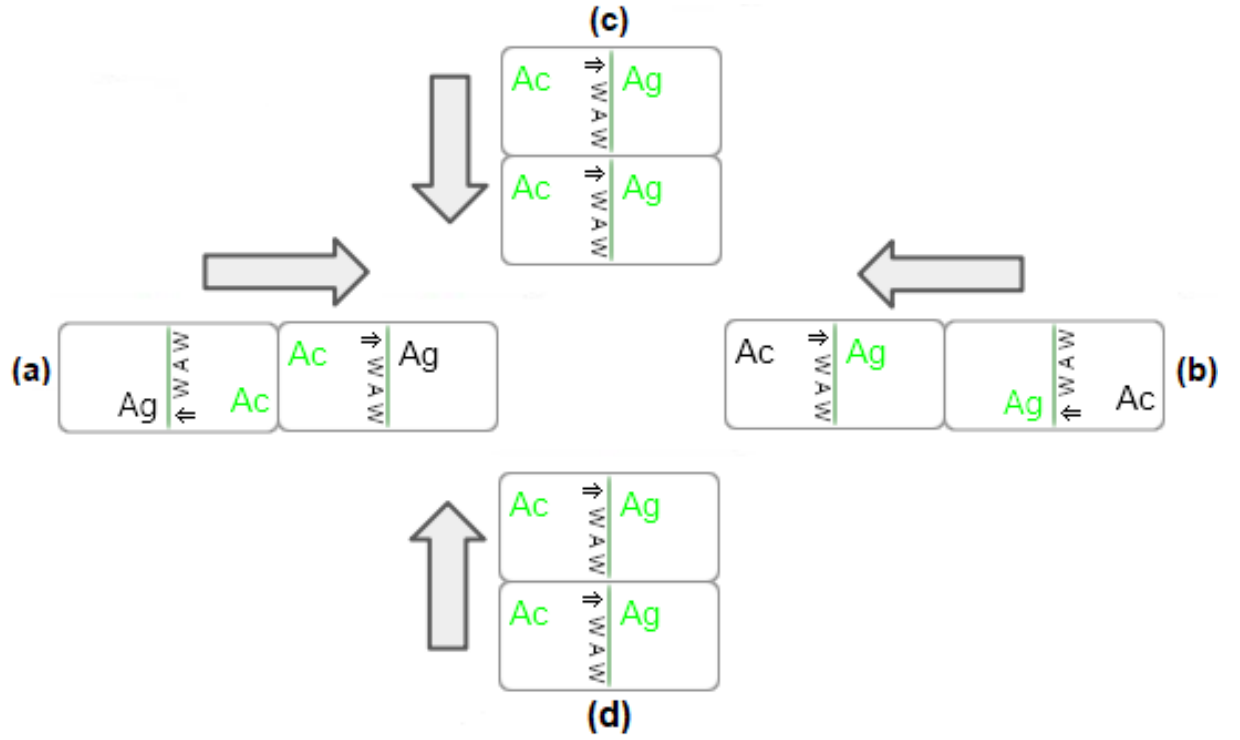


Figure 3.5: Domino tiles combinations through: the left (left-multiplication) (a); or the right (right-multiplication) (b); the top (sum) (c); and the bottom (subtraction) (d).

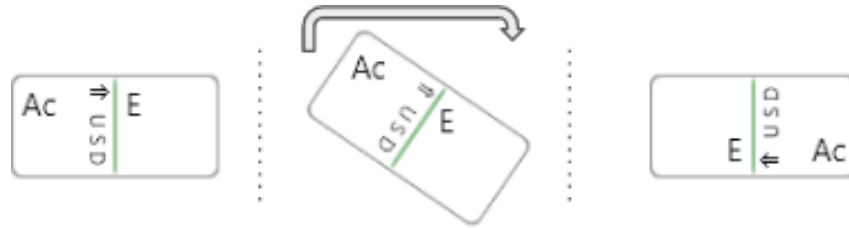


Figure 3.6: Transposition of a domino tile.

domino tiles horizontally (multiplication-merge) in the left as in Figure 3.5.a or the right as in Figure 3.5.b; by approaching two equal³ domino tiles in the top (sum-merge), as in Figure 3.5.c; and by approaching two equal domino tiles in the bottom (subtraction-merge), as in Figure 3.5.d.

As multiplication-merge requires the merging edges to be the same, resembling the dominoes game when leaning two domino tiles, users may need to transpose (reverse the edges) the domino tile to match edges or to take benefit from other perspective during visualization. Figure 3.6 shows such domino tile transposition. The transposition process starts when the user double-clicks the domino tile, issuing a transposition of its underlying matrix (e.g., $\text{USD} \rightarrow \text{USD}^T$).

³Equality of edges and dimensions.

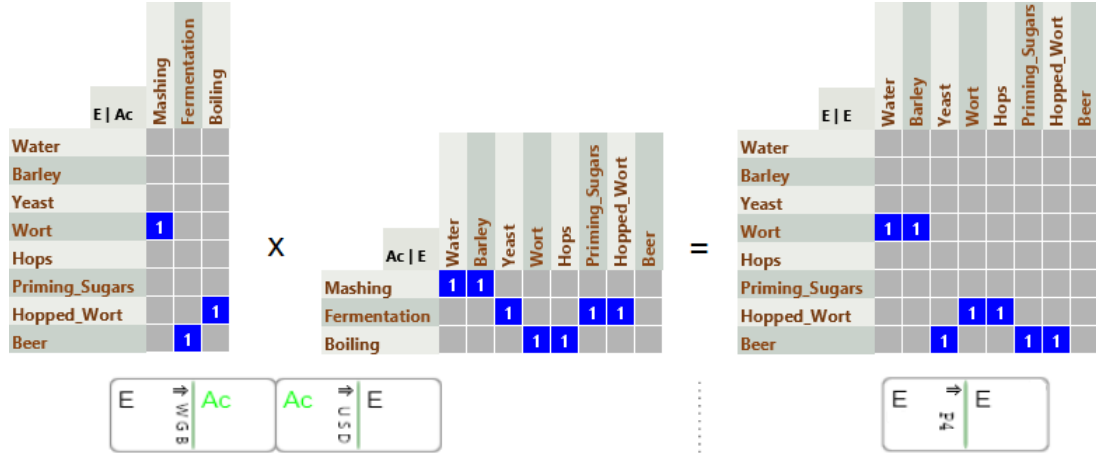


Figure 3.7: Domino tile combination: WGB [E|Ac] \times USD [Ac|E], resulting in a WDF [E|E] domino tile.

When approaching edges horizontally, Prov-Dominoes performs matrix multiplication, deriving a new matrix for the resulting domino tile, as shown in Figure 3.7. The multiplication takes into account: PROV-DM Type match (object of the first domino tile matches the subject of the second) and dimension match (number of columns in the first domino tile matrix equals to the number of rows of the second). Before the multiplication, Prov-Dominoes sorts the columns of the first matrix and the rows of the second. Such a sorting assures that all elements appear in the same order in both tiles for a correct semantic interpretation.

For instance, the example shown in Figure 3.7 consists of a derivation inference [15]. The domino tile in the left-hand side shows entities generated by activities (WGB [E|Ac]), and the domino tile in the right-hand side shows activities that used entities (USD [Ac|E]). When multiplying both, by connecting the activity side, the generated domino tile shows entities derived from other entities (WDF [E|E]). By using the matrix visualization over the new domino tile, we can see that only the generated entities have some cells filled: *Hopped_Wort*, *Wort*, and *Beer*. In the columns we can see the entities that took part in the generation of the row entities: *Hopped_Wort* (row) was derived from *Wort* (column) and *Hops* (column), *Wort* (row) was derived from *Water* (column) and *Barley* (column), and *Beer* (row) was derived from *Hopped_Wort*, *Yeast*, and *Priming_Sugars*. This allows analysis that would otherwise be difficult to make just by looking at the corresponding provenance graph.

Two other inferences of the PROV-DM Constraints can be obtained by combining domino tiles. The combination of USD [Ac|E] and WGB [E|Ac] by approaching the E edges produces a domino tile corresponding to WIB [Ac|Ac] (wasInformedBy) [12]. The

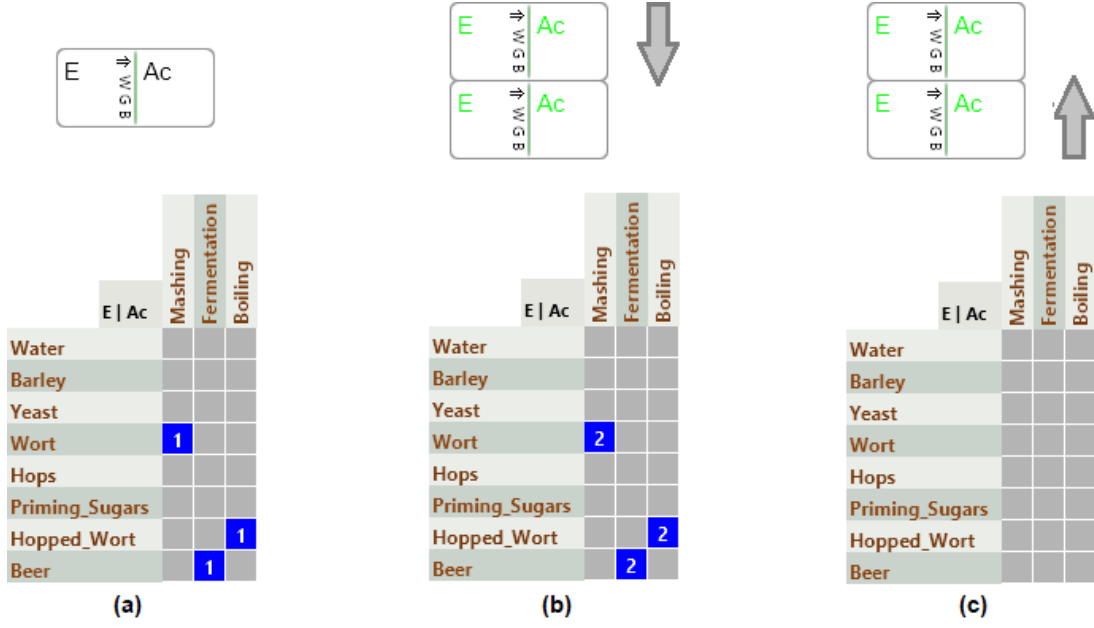


Figure 3.8: WGB [E|Ac] (a); WGB + WGB (b); and (c) WGB – WGB.

combination of WGB [E|Ac] and WAW [Ac|Ag] (wasAssociatedWith) by approaching the Ac edges produces a domino tile corresponding to WAT [E|Ag] (wasAttributedTo) [11]. Once new data arise from combinations, portions of provenance are unveiled, providing a more holistic perspective for provenance analysis.

The sum-merge and subtraction-merge require the edges and dimensions of both domino tiles to be equal. While approaching the domino tiles, the edges' letters become green (edges' letters match) or red (edges' letters do not match). When green, the domino tiles are merged if the dimensions of the approached edges also match, and a new domino tile arises with the same edges and dimensions of the previous two. Under the hood, Prov-Dominoes performs the sum (top-approach) or subtraction (bottom-approach) of the matrices. To ensure the match of rows and columns, Prov-Dominoes previously sorts rows and columns of both domino tiles.

For instance, the example shown in Figure 3.8 illustrates the sum and subtraction of domino tile WGB [E|Ac] (Figure 3.8.a) with itself. Figure 3.8.b exhibits the matrix visualization of the resulting sum and Figure 3.8.c exhibits the matrix visualization of the resulting subtraction.

3.5 Domino Tile Transformations

In this section, we describe transformations available on Prov-Dominoes. These transformations affect only the matrix's content beneath a domino tile, without affecting its external view. Such content-only transformations may reveal implicit information about the data. They are organized as follows:

- **Operations:** affect cell values or matrix dimensions;
- **Filters:** filter (by setting cell value to zero) cells according to some filtering rule; and
- **Sorting:** rearrange cells by moving rows and columns.

In the following list, we briefly summarize the domino tile transformations available in Prov-Dominoes, except for *transitive closure* and *cluster sorting*, which we detail in the following. We refer to $A_{i,j}$ as a cell with row index i and column index j of a generic matrix A that represents a given domino tile:

- *Binarize:* $A_{i,j} \leq 0 \Rightarrow A_{i,j} = 0$ and $A_{i,j} > 0 \Rightarrow A_{i,j} = 1$;
- *High-Pass Filter (HPF):* $A_{i,j} \leq v \Rightarrow A_{i,j} = 0$, where v is a cutoff value;
- *Low-Pass Filter (LPF):* $A_{i,j} \geq v \Rightarrow A_{i,j} = 0$, where v is a cutoff value;
- *Trim:* Eliminates empty rows and columns;
- *Aggregate Rows/Columns:* Reduces rows/columns to one dimension (sum of the rows/columns);
- *Z-score:* Sets standard deviations from the column average on cells sharing the same column;
- *Word on Row/Column:* $A_{i,j} = 0$ if the row/column label of the cell does not match a word (or regular expression); and
- *Sort by Row/Column Group:* Rearranges cells by grouping together non-zero cells in a row/column. The bigger the group, the closer to the top (if a row group) or to the left (if a column group).

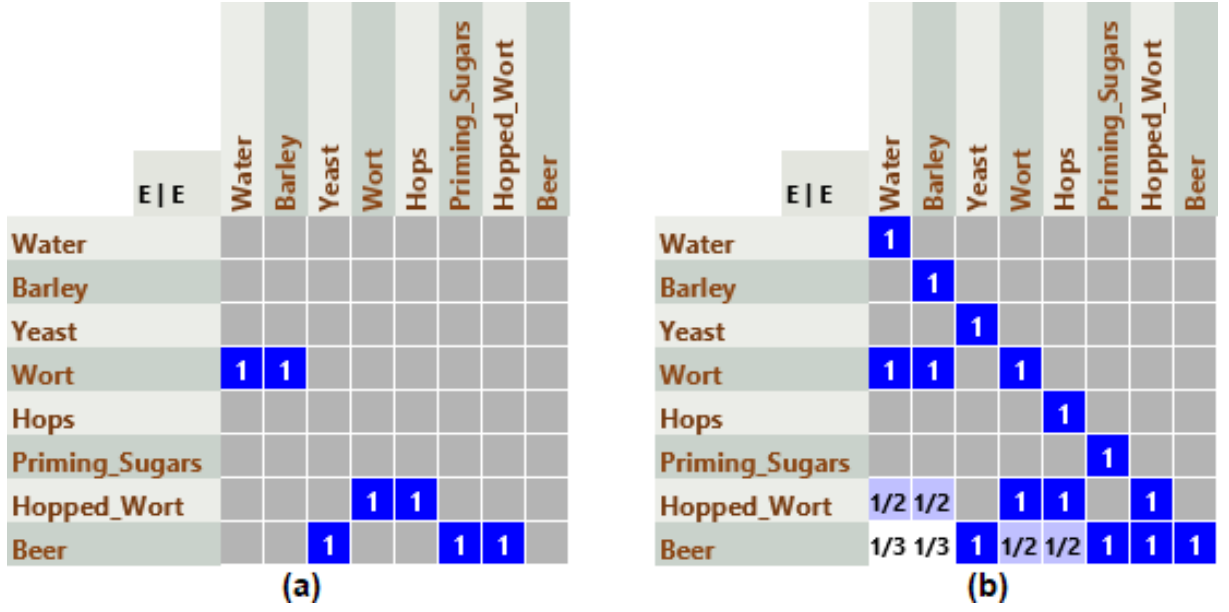


Figure 3.9: The *WDF* matrix visualization (a) and the same matrix after the *transitive closure* operation (b).

Transitive closure is an allowed operation for a double domino tile (i.e., tiles with the same element in both sides) enclosing an adjacency matrix. The *transitive closure* of an adjacency matrix is the reachability⁴ matrix. Typically, generating the *transitive closure* is the process of creating the reachability matrix, where $A_{i,j} = 1$ if a path exists from vertex i to j and $A_{i,j} = 0$ otherwise. Our approach generates an adapted reachability matrix. Instead of setting one when there is a path between i and j , we set $1/n$, where n represents the number of steps necessary to get from i to j . This provides an intuitive visualization of node's distance, as the shades of blue are lighter for nodes distant from each other. Moreover, we assume each vertex can reach itself in one step.

Consider the example from Figure 3.9, where Figure 3.9.a shows the matrix of the WDF $[E|E]$ domino tile produced from our guiding example. Figure 3.9.b shows the same matrix after the *transitive closure* operation. In Figure 3.9.b, we can see that the last row (*Beer*) is transitively derived from all ingredients of the beer provenance, as expected. The blue cells indicate ingredients that were participating in the last step of the beer production. The cells in shades of blue indicate indirect influences to the beer production. For instance, because *Water* and *Barley* were the first ingredients in the beer production, they have lighter blue shades. In fact, they are 3 steps distant to the *Beer* entity: (*Water*, *Barley*) \rightarrow *Wort*, *Wort* \rightarrow *Hopped_Wort*, and *Hopped_Wort* \rightarrow *Beer*.

Furthermore, the *transitive closure* operation enables another inference of the PROV-

⁴In graph theory, reachability refers to the ability to get from one vertex to another within a graph.

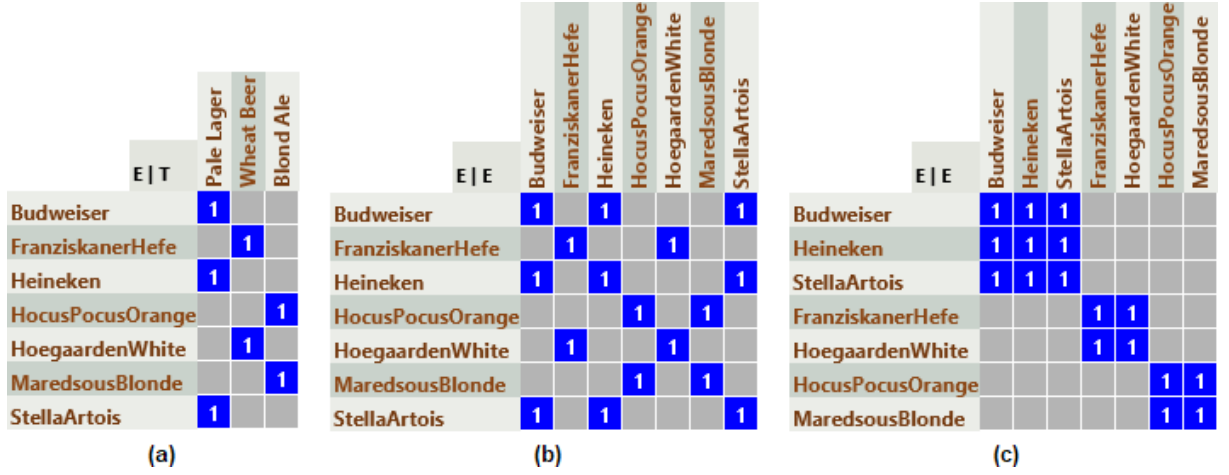


Figure 3.10: $[E|T]$ representing some beers and their types (a). $[E|T] \times [E|T]^T = [E|E]$ (b). $[E|E]$ after *Cluster Sorting* (c).

DM Constraints, the delegation inference [14]. For instance, in Figure 2.2, there is no connection between *Bill* (an *agent*) and *Mashing* (an *activity*). The delegation inference states that, because *Bill* has one delegate (*John*), he is also associated with the activity (*Mashing*) of his delegate. If we combine WAW $[Ac|Ag]$ with the *transitive closure* of AOB $[Ag|Ag]$ (actedOnBehalfOf) by approaching the *Ag* edges, we obtain a new domino tile that is the same WAW $[Ac|Ag]$, but considering the delegation inference. Thus, the cell of *Mashing* and *Bill*) is set to one.

The *Cluster Sorting* is a rearrange method that brings cells sharing the same row or column closer, leaving empty cells grouped (to the right above the diagonal or the left below the diagonal). In Figure 3.10.a, we show an *entity-type* $[E|T]$ domino tile representing some beer brands and their types. In Figure 3.10.b, we have the resulting matrix of $[E|T] \times [E|T]^T = [E|E]$, a domino tile representing pairs of beer brands (cells) sharing the same type. After the rearrange produced by the *Cluster Sorting*, we obtain the matrix of Figure 3.10.c. We can see the block formations over the diagonal. The first 3x3 block groups the pale lager beers, the second 2x2 block groups wheat beers, and the last 2x2 block groups blonde ales.

Chapter 4

Implementation

In this chapter, we discuss the Prov-Dominoes architecture and its GUI. We discuss the three layers of the architecture and how each of its components orchestrate together to achieve the goals mentioned in Section 1.2. Finally, we overview the tool’s GUI, highlighting its main parts and features.

4.1 Prov-Dominoes Architecture

The architecture of Prov-Dominoes has three layers: *data layer*, *processing layer*, and *presentation layer*. The *data layer* is responsible for reading the supported input files and parsing the data in the files to matrices for domino tiles assembling. The *processing layer* provides the algorithms for matrix combinations and transformations, both for GPU and CPU. These two layers are depicted in Figure 4.1 and are described in this section. The *presentation layer* is discussed later, in Section 4.2.

The *data layer* has three inner components: *Prov-Matrix*, *Domino Tile Parser*, and *Script Processor*. The *Prov-Matrix* component is in charge of converting PROV-N expressions into matrices, one matrix per distinct PROV-DM Relation stated in the PROV-N file, as mentioned in Section 3.1. After the matrices are built, they are sent to the *Domino Tile Parser*. The *Domino Tile Parser* component is responsible for assembling the domino tile, both in terms of its matrix content and its non-graphical structures, such the labels for the matrix and dimension types. After this process, the domino tile data structure is available for the *presentation layer*, where the domino tiles are drawn and manipulated.

The *Script Processor* component is responsible for building and managing the history of commands in a tree data structure. Each exploratory-related action performed by

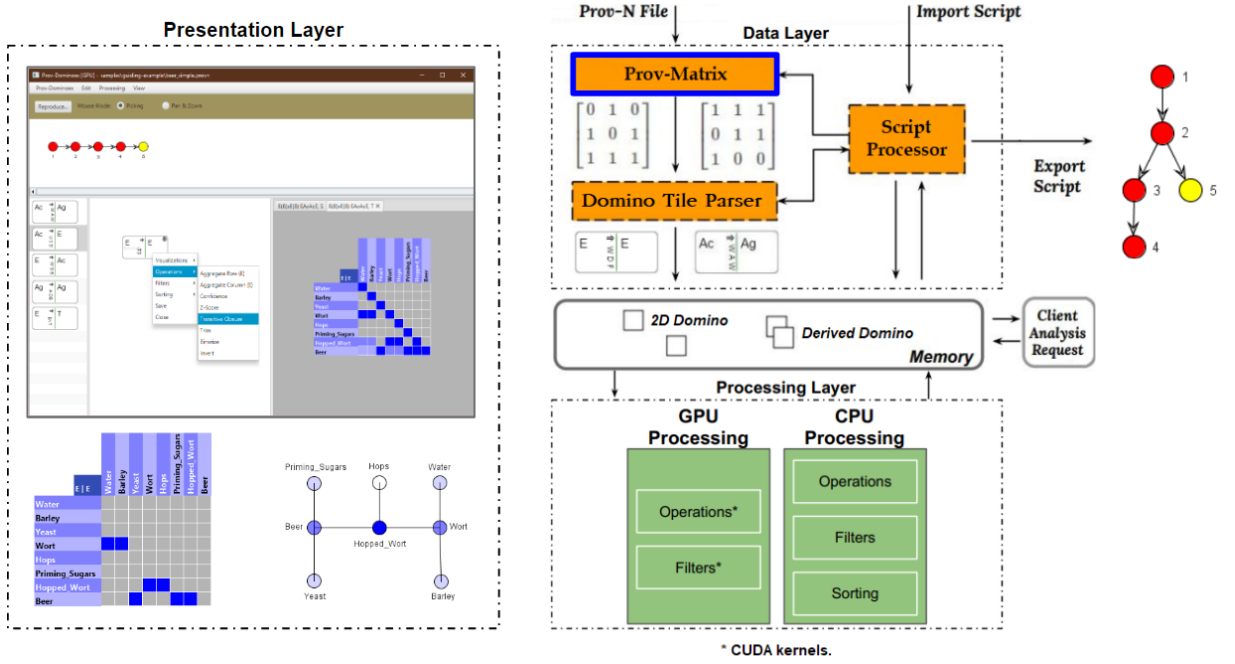


Figure 4.1: Prov-Dominoes' architecture.

the user is referred as a command¹. Such command history functions as the provenance of the derived domino tiles, enabling the user to keep track and navigate through their explorations. The *Script Processor* enables the user to export the commands that led to the derived domino tiles (i.e., their provenance) to an *Exploration Provenance Script* (EPS). Similarly, the *Script Processor* allows the user to import the EPS to reproduce previous explorations.

In Figure 4.2, we show an example of an EPS. The first commands *LOAD*, *ADD*, *MOVE*, and *TRANSPOSE* are represented by nodes 1, 2, 3 and 4. The command of node 2 ($p1 = \text{ADD}(\text{USD})$) has an identifier: “p1”. The “p1” identifier indicates that the *ADD* command produced a domino tile (a copy of *USD* as result of $\text{ADD}(\text{USD})$) that was stored in “p1”, allowing further commands to access such result by referring to “p1”, as in the forth command: $\text{TRANSPOSE}(p1)$. After the forth command, the next command $\text{UNDO}(n = 2)$ represents two undo performed in sequence, invalidating the n last commands and moving backward n nodes in the history tree. Commands performed after *UNDO* will imply a new branch created in the history tree, as in command $p2 = \text{ADD}(\text{WGB})$, attributed to node 5.

The *processing layer* consists of two matrix processing components, in CPU and GPU, that operate in a mutually exclusive processing mode. They are responsible for processing matrix-related transformations: operations, filters, and sorting. The CPU processing

¹A list of all commands is available at Appendix B.

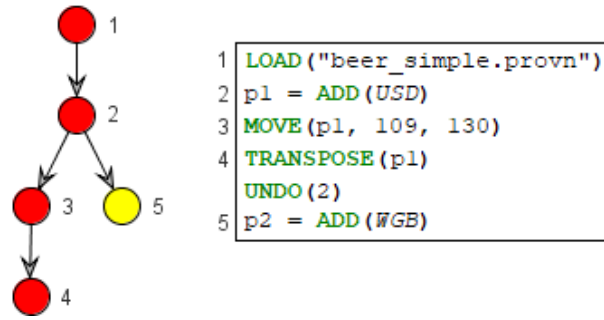


Figure 4.2: Exploration Provenance Script.

component uses the open-source java library *Linear Algebra for Java* (LA4J)². The library provides Linear Algebra primitives (matrices and vectors) and algorithms for Java. The GPU processing component is implemented in *Compute Unified Device Architecture* (CUDA)³, a parallel computing platform and API model created by NVIDIA. The asterisks in Figure 4.1 indicate the matrix-related transformations that were implemented in CUDA:

- Operations: addition, subtraction, transposition, multiplication, transitive closure, z-score, and binarization;
- Filters: HPF and LPF.

The purpose of GPU usage is to enable faster processing when dealing with large provenance data, where CPU may be limited and not perform satisfactorily. All transformations are executed in CPU, if GPU is not available in the computer.

Aiming at enriching the current architecture, we considered to develop a plugin to import log application files as provenance for analysis in the tool. We were able to complete functions for converting logs into PROV-N expressions, however it was not incorporated into the architecture due to time and scope restrictions. Nevertheless, we released such converter as a separate Java API called **log2prov**⁴.

4.2 Prov-Dominoes GUI

The Prov-Dominoes GUI uses Java FX for its graphical components and has four main panels (see Figure 4.3): Command History Tree, Domino Tiles List, Canvas, and Visualization Tabs. The **Command History Tree** lies in the top area, as shown in Figure 4.3.a.

²<http://la4j.org>.

³<https://developer.nvidia.com/cuda-zone>.

⁴<https://github.com/gems-uff/log2prov>.

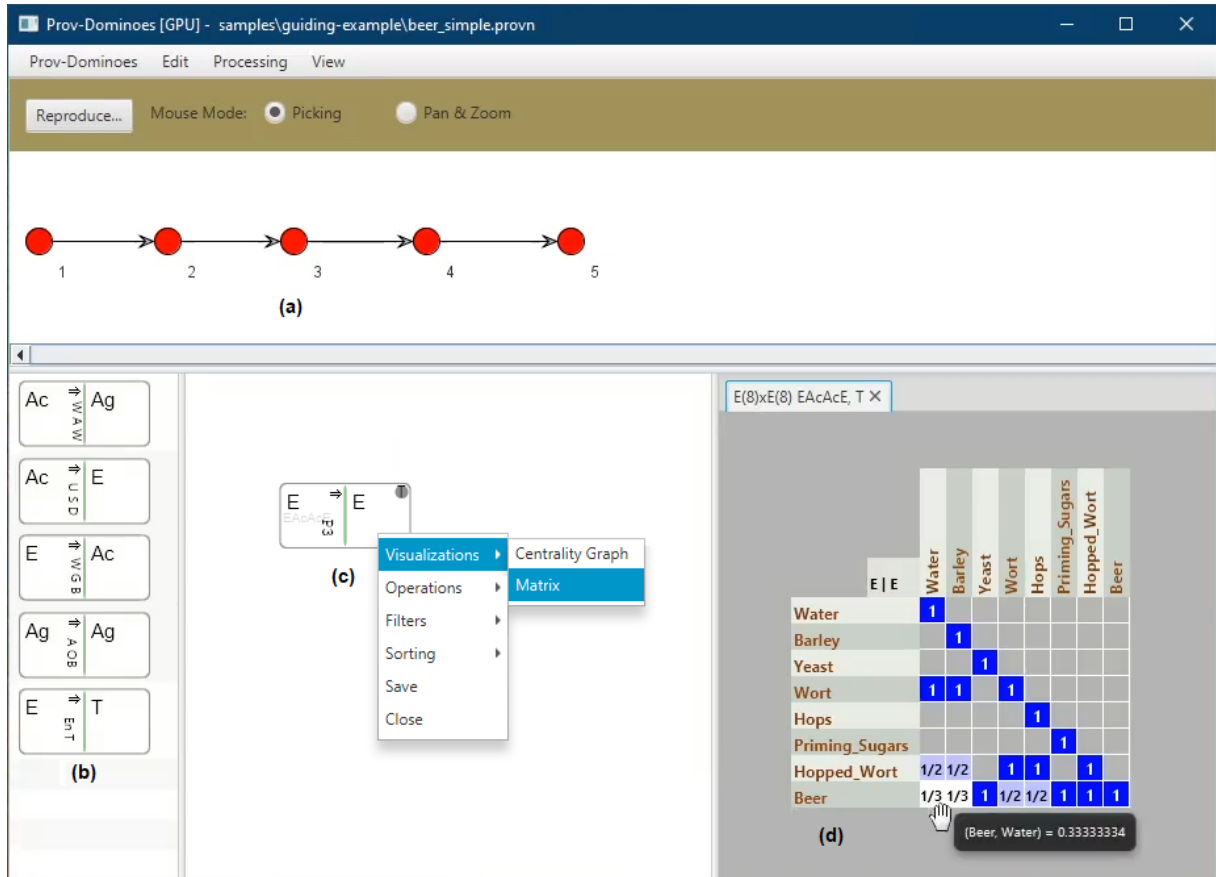


Figure 4.3: Prov-Dominoes GUI: Command History (a); Domino Tiles List (b); Canvas (c); and Visualization Tabs (d).

The domino tiles assembled from the provenance files or EPS are exhibited in the **Domino Tiles List**, bottom left in Figure 4.3.b. By double-clicking a domino tile in the Domino Tiles List, the clicked domino tile goes to the **Canvas** in Figure 4.3.c, where combinations and transformations are performed.

Inspired by the History Tree in VisTrails [9], new nodes arise in the Command History Tree as the user manipulates domino tiles in the Canvas. A node in the Command History Tree can be selected to go back to a particular state. The “Reproduce” button resets the Domino Tiles List and Canvas and performs a sequential execution of commands from node one to the selected node.

Right-clicking a domino tile in the Canvas drops down a context menu, as illustrated in Figure 4.3.c. The *Matrix* and *Centrality Graph* visualizations are available under the sub-menu *Visualizations*. When selected, these visualizations are presented in a **Visualization Tab**, as shown in Figure 4.3.d.

To better illustrate the features of Prov-Dominoes, let us consider an hypothetical scenario where Sofia wants to join her friend John (agent of our guiding example in

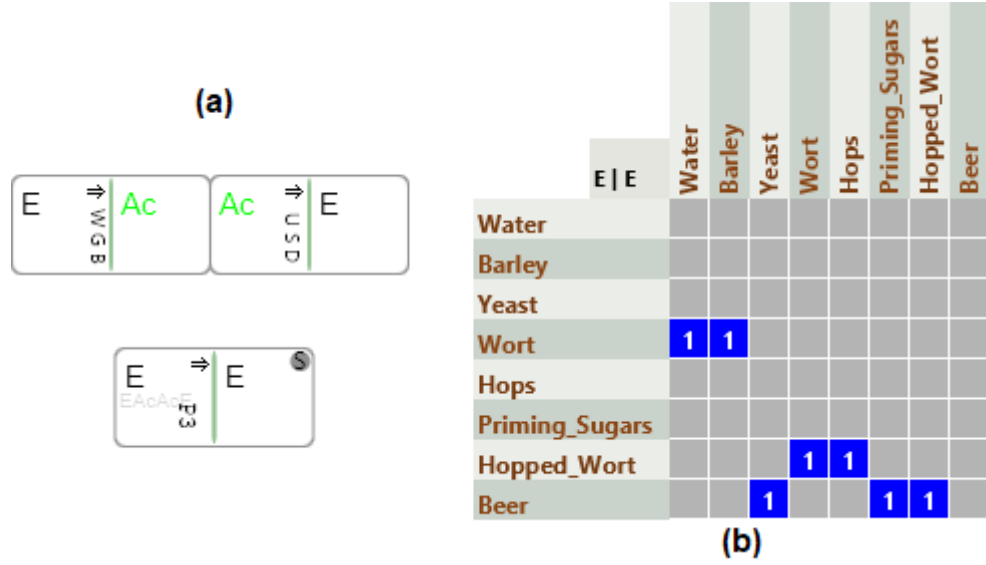


Figure 4.4: WGB combined with USD (in the top), producing WDF (in the bottom): $[E|Ac] \times [Ac|E] = [E|E]$ (a); and WDF matrix content (b).

Section 2.1.1 to also learn with Bill how to make beer. As a test, Bill gives the Beer provenance to Sofia and ask her to identify which other ingredients are present in the hopped wort, besides wort and hops.

After loading the Beer provenance in Prov-Dominoes, Sofia realizes that there is no domino tile relating the ingredients (entities) to each other (i.e, a domino tile with E on both edges), as depicted in Figure 4.3b. Thus, she decides to produce such domino tile by inserting and combining WGB with USD in the Canvas. The combination produces a domino tile with entities on both edges, as shown in Figure 4.4.a.

Sofia decides then, to observe the underlying matrix of the produced WDF domino tile. She sees the matrix contents as in Figure 4.4.b and realizes that she is seeing only direct derivations related to each ingredient. For example, the row *Hopped_Wort* is a direct derivation of the columns *Wort* and *Hops*. However, she realized that indirect derivations are missing. In his question to Sofia, Bill already mentions *Wort* and *Hops* as ingredients present in the *Hopped_Wort*.

Sofia realizes that she needs to check if any other ingredients in the columns can reach *Wort* or *Hops* in the rows, implying that she needs the corresponding reachability matrix from the one in Figure 4.4.b. In order to produce such reachability matrix, she applies the *transitive closure* operation as in Figure 4.3.c, obtaining the reachability matrix displayed in Figure 4.3.d. Now, in the row *Hopped_Wort*, two new ingredients appear as indirect participants of *Hopped_Wort*: *Water* and *Barley*, the answer to Bill's question to Sofia.

Besides loading the provenance file, Sofia executed four steps to get to the answer she was seeking. Figure 4.3.a lists all the steps performed by Sofia. The first one was the provenance loading. Second and third, the insertion of WGB and USD into the canvas for manipulation. The forth step was the combination between the WGB and USD. The fifth and last step was the transitive closure operation. Moreover, Sofia can share her findings by exporting (option “Export to script...” in “Prov-Dominoes” menu) her explorations.

Chapter 5

Evaluation

In this chapter we present and discuss the research questions over which we evaluate Prov-Dominoes. Our assessment first focus on an effectiveness evaluation, addressed over three distinct case studies. After that, we present an efficiency evaluation through large data processing, which considers thousands of relations collected from Twitter.

The following Research Questions are considered:

RQ1: How effective is Prov-Dominoes in supporting interactive exploration of provenance data? As discussed in Chapter 1, making explicit any implicit provenance information is an aspect of provenance analysis. Thus, we considered the capability of uncovering implicit data (i.e., making it explicit) an effective aspect in supporting interactive exploration of provenance data. Moreover, as provenance data may be abundant in explicit information, we considered the capability of providing holistic perspectives over (possibly abundant) information another effective aspect to guide explorations. Additionally, we consider concise analysis a requirement, whose benefits would integrate both strict (implicit and explicit) and holistic data, altogether compounding effective support for interactive explorations. Thus, we broke down effectiveness into three sub-questions:

RQ1.1: How Prov-Dominoes uncovers implicit information? Provenance data may have implicit information buried into the data, hindering its perception. Uncovering such information may enrich the interactive exploration of provenance data.

RQ1.2: How Prov-Dominoes provides a holistic perspective? Similarly, big picture analyses, provided by a holistic perspective of the data, may contribute to provide a wholesome comprehension of the data domain. We analyze if the explorations performed using Prov-Dominoes are capable of extracting such holistic perspectives.

RQ1.3: How Prov-Dominoes supports concise analysis? We assess if the

matrix and *eigenvector centrality* graph visualizations provided by Prov-Dominoes are capable of integrating strict and holistic information, thus assisting the explorations. Furthermore, as discussed in Chapter 1, analyzing provenance graphs start to become an overwhelming task as the number of edges and vertices grows. Providing concise ways to visualize graphs may not only benefit the analysis on larger graphs, as discussed in Section 3.3, but also potentially assist pattern identifications regarding the arrangements of cells in a matrix. Moreover, the *eigenvector centrality* graph may benefit the analysis by per-type observations. For instance, first analyzing the centrality of agents, then entities, and then activities.

RQ2: How efficient is Prov-Dominoes when running in GPU in comparison to CPU? In general, GPU applications are expected to perform faster than its counterpart in CPU, due to its parallel architecture. However, it is not straightforward to know from which data volume GPU processing starts to become faster than CPU (i.e., the point where data processing hides memory transfer latency) for a particular application, and the magnitude of the speedup (CPU time / GPU time). We analyzed how faster is Prov-Dominoes in the GPU mode in comparison to the CPU mode when processing thousands of relations and from which data volume on it becomes faster.

In the following sections, we describe the corpus and discuss the results of our explorations, first for the effectiveness evaluation (**RQ1**), then for the efficiency evaluation (**RQ2**).

5.1 Effectiveness

In this section we present three distinct case studies as an attempt to answer **RQ1.1**, **RQ1.2**, and **RQ1.3**. First, we describe the materials and methods of the case studies, and then we present and discuss the results. In the end, in Section 5.1.3, we summon some exploratory practices frequently observed in the case studies.

5.1.1 Materials and Methods

We selected three different case studies in order to observe how Prov-Dominoes supports exploratory analyses in different domains. For each case study, we designed exploratory tasks for information extraction, considering their domains.

The **first case study (Animals)** used data from a UCI [27] dataset about animals

and was selected because of the general knowledge readers may have about the domain, easing the understating of potential findings. Despite not yet a provenance, we present such simple dataset as an instructive study paving the way to the other two provenance case studies. The exploratory tasks focus on identifying activity patterns among animal classes and outlining the ruling activities of each animal class.

The selected Zoo [30] dataset contains 101 animal entries associated with 15 Boolean attributes and two categorical attributes. The first categorical attribute refers to the animal class: mammal, bird, reptile, fish, amphibian, insect, or other. The second categorical attribute refers to the number of animal’ legs: 0, 2, 4, 5, 6, or 8. For matrix indexing, we unfolded the categorical attributes into separated Boolean values. The remaining attributes refer to animal activities.

When parsing the dataset to the PROV-N notation, we represented the animals as agents, their classes (animal type) as *agent-types*, and their actions as activities. We used the *wasAssociatedWith*(*Ac*, *Ag*) PROV-N expression to represent that animal activities (*Ac*) were associated with animals (*Ag*). We used the “prov:type” attribute (e.g., `agent(pitviper, [prov:type=“reptile”])`) to relate an animal to its class. As result, Prov-Dominoes generated two domino tiles: WAW [*Ac*|*Ag*] and *agent-type* [*Ag*|*T*].

The **second case study (Workflow)** uses provenance from a VisTrails’ workflow called “Head.” We wanted to lean on a provenance generated by a Workflow Management System (WfMS) to check if our explorations could contribute to a better understanding of the provenance. The exploratory tasks aim at understanding which activities and parameters were essential to the workflow execution results. For this case study, we adopted member checking by inviting the workflow’s author to provide feedback about our explorations.

The chosen Head Workflow gathers data derived from “The Visible Human Project” [1] corresponding to a person’s head and renders its bones and skin as a volumetric image. When exporting the Head Workflow from VisTrails to PROV-N, apparently only the executions of the last exploration is available. The last exploration in the sample is called “volume rendering,” that is the one we considered for analysis. The modules used in the execution are based on the Visual Toolkit (VTK) [52] and depicted in Figure 5.1.

The exported provenance maps the modules as activities, the inputs and outputs as entities, and the executors of the workflow as agents. The provenance visualization of the workflow in VisTrails focuses on module (activities) connections, as shown in Figure 5.1. Each edge connecting two modules indicates that the output (*entity*) produced by the

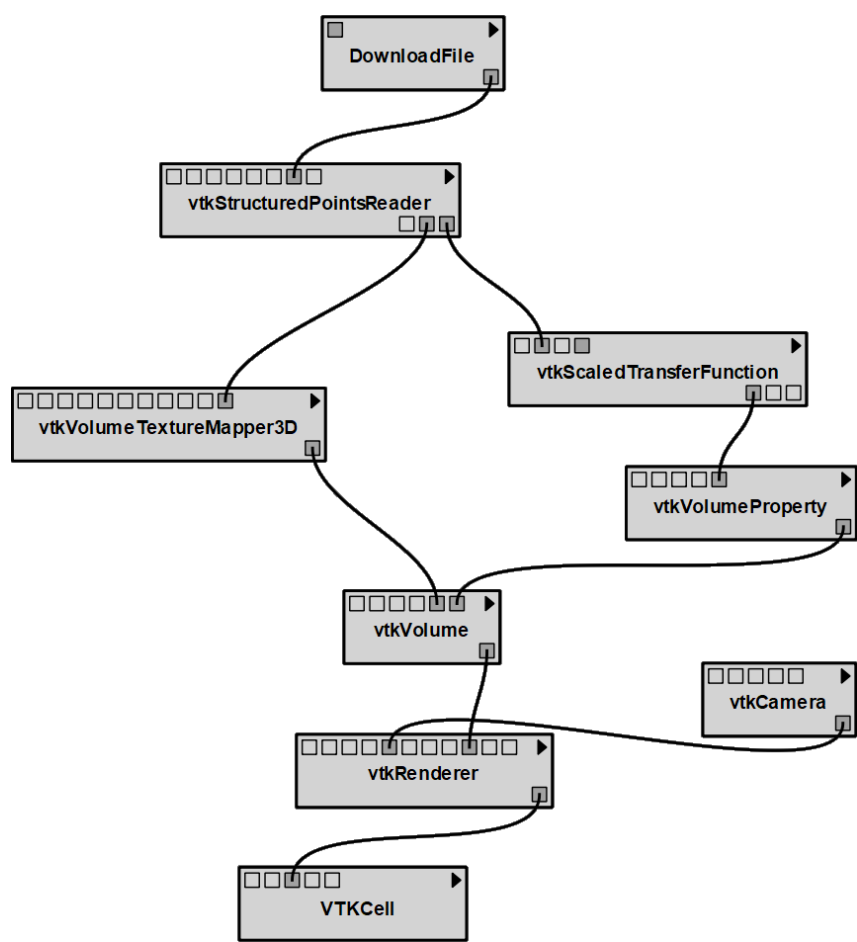


Figure 5.1: Head Workflow and its executed modules.

source module was used as input (*entity*) by the target module. However, we do not see such entities in Figure 5.1, as they are kept implicit. Furthermore, additional parameters one module may have, are also kept implicit. As the “volume rendering” exploration was executed only by the workflow’s author, we refrain from *agent* analysis in this case study.

The **third case study (Smart Home)** uses provenance captured from a smart home service called MiJia¹. We chose such a case study as an example of the potential of provenance on smart devices and the internet of things. We downloaded the provenance captured by a user of such smart home service from ProvStore², a free service for storing, viewing, and collaborating on provenance data. The exploratory tasks investigate which activities are central to the smart home service’s functioning and which agents are behind its operation.

The MiJia (Mi Home) Smart Home Service consists of a series of Xiaomi³ smart devices operating together in a building. The smart devices communicate with a multi-functional gateway responsible for integrating them. The user can control and interact with smart devices through a MiJia app on her smartphone. The data generated by the service is recorded as PROV-N, where device data are represented as entities, commands issued to the devices as activities, and devices as agents. The captured provenance has expressions about four devices: a temperature sensor (“Sensor1”), a humidity sensor (“Sensor2”), and two unspecified devices (“Device1” and “Device2”), as shown in Figure 5.2. Additionally, there are two agent-types: *prov:SoftwareAgent* and *prov:Person*. The former encompasses the agents “Server” (operating on the cloud) and “Application” (operating on the user’s smartphone), and the latter encompass the agent “User”.

5.1.2 Results and Discussion

In this section, we detail how transformations and visualizations articulated together during exploratory tasks in each case study.

In the **Animals case study**, to address the first exploratory task (identifying activity patterns among animal classes), we combined domino tiles aiming to relate animal classes and their activities, as follows: $[Ag|T]^T \times [Ac|Ag]^T = [T|Ac]$ (classes and their activities). After the combination, we rearranged the cells for pattern analysis. The matrix visualization of the resulting domino tile is shown in Figure 5.3.

¹<http://home.mi.com/index.html>.

²<https://openprovenance.org/store/documents/2148>.

³<https://www.mi.com>

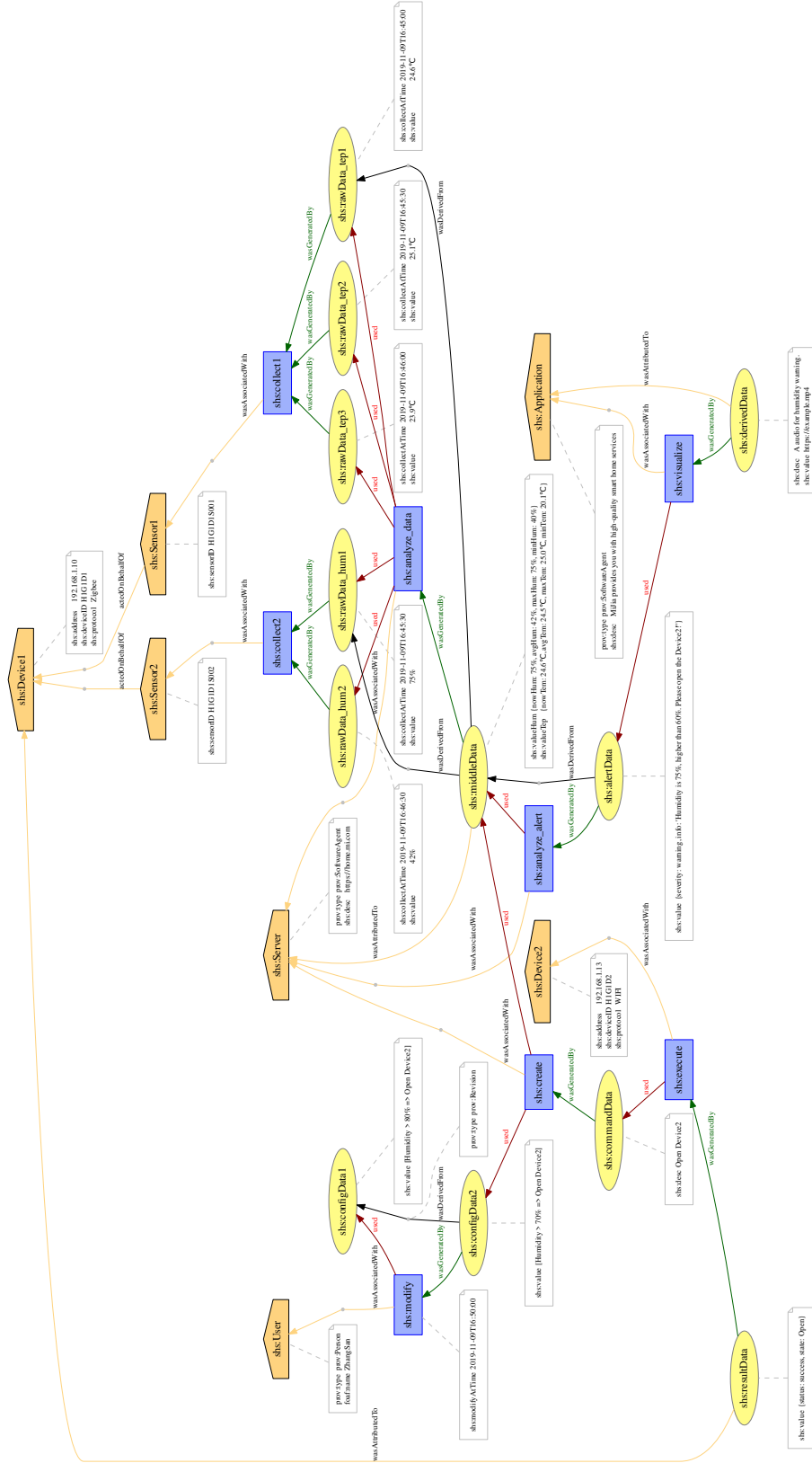


Figure 5.2: Provenance graph taken from ProvStore.

	T Ac	lays_eggs	hunts_prey	handles_tail	breathes_air	feeds_underwater	produces_venom	grows_to_catsize	emits_nerve_signals	allows_domestication	moves_with_0_legs	moves_with_4_legs	grows_teeth	flies	moves_with_6_legs	moves_with_2_legs	produces_hair	aims_fins	moves_with_8_legs	produces_feathers	produces_milk	moves_with_5_legs
mammal		1	22	35	41	6	32	41	8	3	31	40	2		7	39	4				41	
other		9	1	3	6		1			4	1			2								1
fish		13	9	13	13	4	13	1	13		13							13				
bird		20	9	20	20	6	6	20	3				16		20					20		
reptile		4	4	5	4	1	2	1	5		3	2	4									
amphibian		4	3	1	4	4	1		4			4	4									
insect		8			8		2			1				6	8		4					

Figure 5.3: Classes x Activities ($[Ag|T]^T \times [Ac|Ag]^T = [T|Ac]$).

The first two vertical lines of the matrix in Figure 5.3 indicate that all classes lay eggs and hunt, although mammals and insects rarely lay eggs and hunt, respectively. From now on, we discuss highlights per column, from left to right, and top-down inside the column. In the following column, we highlighted a cell indicating that only insects do not handle tail. The findings in the remaining highlighted cells are orderly summarized as follows:

- Only fishes do not breathe air;
- Only insects do not feed underwater;
- Mammals and birds do not produce venom;
- Mammals rarely fly;
- Only mammals and fishes aim fins;
- Only birds produce feathers; and
- Only mammals produce milk.

To address the second exploratory task (outlining the ruling activities among animal classes), we applied the *Z-score* operation for dispersion analysis to the prior generated $[T|Ac]$ domino tile. As the cell values represent standard deviations, negative cells indicate below average, zero indicates average, and positive cells indicate above average. Then, we applied the *Binarize* transformation to obtain the ruling activities of each animal class. After this transformation, the remaining filled cells represents activities that occur on

	Ac T	mammal	fish	bird	insect	amphibian	reptile
lays_eggs		1	1	1	1	1	1
emits_nerve_signals		1	1	1		1	1
breathes_air		1		1	1	1	1
handles_tail		1	1	1			1
grows_teeth		1	1			1	1
moves_with_4_legs		1				1	
flies				1	1		
feeds_underwater			1			1	
moves_with_0_legs			1				
produces_hair		1					
produces_milk		1					
moves_with_6_legs					1		
moves_with_2_legs				1			
hunts_prey							1
produces_feathers				1			
aims_fins			1				
grows_to_catsize		1					

Figure 5.4: Ruling Activities x Classes.

average or above average in an animal class. To eliminate the class “other”, we filtered (*Word on Column*) the columns with a regular expression containing all classes except “other”, and trimmed the resulting *Z-score* domino tile. Finally, we grouped (*Sort by Row/Column Group*) the ruling activities (rows) to ease visualization according to the number of shared activities among classes (columns).

Figure 5.4 shows the resulting domino tile of our second exploration. Above the first black line, we can see the ruling activities shared among five classes. The ruling activities shared among four classes are above the second black line. Above the third, we see the ruling activities shared among two classes. The remaining ruling activities are class-specific. In the columns, we can see that the classes mammal and insect are the ones with more (8) and less (4) ruling activities, respectively. Two of the 15 original Boolean attributes do not appear in the table: “produces_venom” and “allows_domestication.” Moreover, from the six-leg types (0, 2, 4, 5, 6, and 8), two types are absent in the table: movements with 5 and 8 legs.

RQ1.1. *How Prov-Dominoes uncovers implicit information?*

In the provenance, there were no data directly relating activities to animal classes. Such relation was made explicit by exploring the combination between activities and agent-types (animal classes): $[Ag|T]^T \times [Ac|Ag]^T = [T|Ac]$. By unveiling the implicit data about activities and animal classes, further explorations, such as ruling activities, became possible.

RQ1.2. *How Prov-Dominoes provides a holistic perspective?*

The matrix of Figure 5.4 provided us with a big picture view of animal classes and their ruling activities, contributing to a clear comprehension of the domain.

RQ1.3. *How Prov-Dominoes supports concise analysis?*

The concise matrix visualizations used in our explorations enabled pattern identifications such as vertical lines indicated in Figure 5.3 and dense/sparse regions as above/below the third black line in Figure 5.4, respectively. Moreover, the gray color and blue shades in Figure 5.3 smoothly guided our attention to absent, low, or high occurrences in the matrix, easing the analysis.

In the **Workflow case study**, we decided to focus on entities (input parameters and outputs) to address the exploratory task of understanding which activities and parameters were essential to the workflow, as they represent connecting bounds between activities (modules). Thus, we explored derivation inference to relate entities to each other: $WGB [E|Ac] \times USD [Ac|E] = WDF [E|E]$ [15]. We then applied the *transitive closure* operation in the WDF domino tile to investigate dependencies amongst entities. We filtered the rows of the resulting domino tile to exhibit only outputs. Thus, in the rows, we have outputs, and in the columns, we have entities (inputs and outputs) that took part in the output (row) generation. Finally, we trimmed the domino tile to eliminate empty rows and columns. The EPS expressing the resulting domino tile is shown in Figure 5.5.a and its resulting matrix is shown in Figure 5.5.b.

The two vertical lines in Figure 5.5.b indicate column entities that took part directly or indirectly in almost any output, except for the “vtkCamera” output. The left vertical line is the output of the download module, which gathers the dataset from the Visible Human Project. The right vertical line is the download URL parameter where the dataset is located. Such entities suggest two conditions without which the workflow could not proceed: internet connection and correctness of the URL.

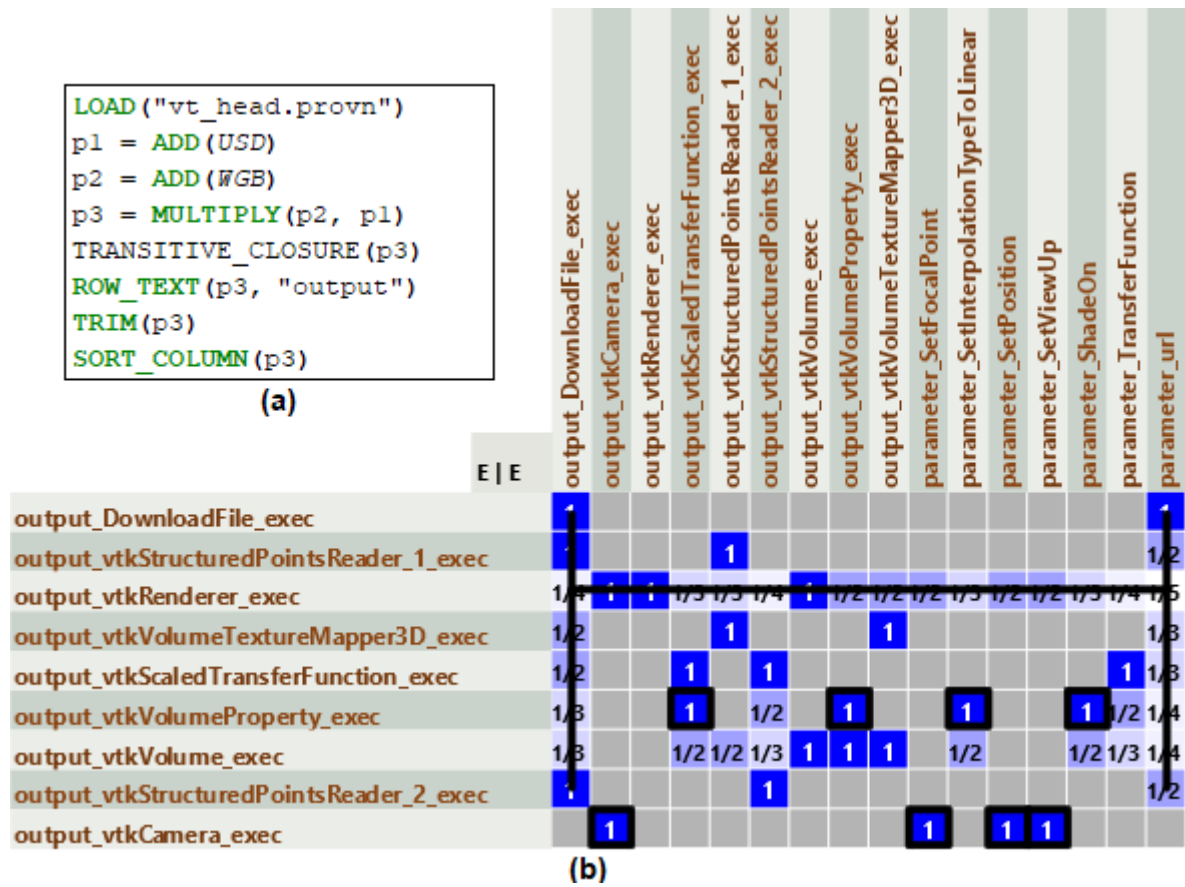


Figure 5.5: EPS of the analysis on entities (a); Resulting domino tile and matrix: outputs in the rows, and inputs and outputs in the columns. The columns represent entities that took part in the output (row) generation (b).

The horizontal line in Figure 5.5.b refers to the most indirectly dependent output, which is the output of the “vtkRenderer.” Such output depends on all inputs and intermediate outputs, making it sensitive to errors propagated by other parameters or intermediate activities. Thus, errors or undesired results detected in the output of the “vtkRenderer” module are harder to debug. Such dependencies also suggest that the “vtkRenderer” module is an aggregation activity, the final destination to prior derived entities, where they may be assembled or processed together, potentially representing a macro aspect or goal of the workflow.

Moreover, in the matrix of Figure 5.5.b, we counted the rows with more direct dependencies (blue cells, cell value = 1), and two rows arose: the outputs of “vtkCamera” and “vtkVolumeProperty.” Both rows have four direct dependencies. Direct dependency is a measure of coupling, useful to determine how complex the testing of various parts of a design are likely to be [19]. According to Yadav and Khan [57], coupling increases the complexity of the system, which makes the system difficult to understand and maintain. Such coupling remarks suggest that “vtkCamera” and “vtkVolumeProperty” are activities complex to maintain and understand.

In order to support our analysis, we interviewed the workflow’s author and asked her to evaluate whether our analysis correctly highlights relevant parts of the workflow. The answer was the following: *“Yes, because the analysis highlights the main components of the workflow. The “vtkRenderer” module is the one that draws the “Head”. The module where I specify how the elements will be drawn is the “vtkVolumeProperty”. The “vtkCamera” sets an observer on the scene and tells it where to look. Without the “vtkCamera”, it would not be possible to see what the “vtkRenderer” module drew.”*

Then, we asked if the analysis was useful. The reply was: *“I considered it extremely useful, especially to see the parameters actually used by the modules and have information on how they are influencing the workflow’s execution. In the workflow view of VisTrails, we can see only the modules. Also, in the matrix, it is easier to recognize coupled modules than in the workflow view. Even in this case, as it is a relatively simple workflow! Imagine one with more connections. It would be much harder to perceive this information in the workflow view!”*

RQ1.1. *How Prov-Dominoes uncovers implicit information?*

In the provenance, there was no data directly relating entities (inputs and outputs) to each other. Such relation was made explicit by the derivation inference: $\text{WGB } [E|Ac] \times \text{USD } [Ac|E] = \text{WDF } [E|E]$. By unveiling *entity* derivations, further explorations on entity dependencies became possible.

RQ1.2. *How Prov-Dominoes provides a holistic perspective?*

As stated by the workflow’s author, the analysis on the matrix of Figure 5.5 was capable of highlighting the main components of the workflow, contributing to a wholesome understanding of the workflow execution.

RQ1.3. *How Prov-Dominoes supports concise analysis?*

The concise matrix visualization used in our explorations enabled pattern identifications such as the horizontal and vertical lines indicated in Figure 5.5, easing the analysis. Moreover, the blue shades in the rows indicated direct dependencies, assisting the dependency analysis. Finally, as stated by the workflow’s author, the visualization provided by Prov-Dominoes is more scalable for workflows with many connections than a plain workflow view.

In the **Smart Home case study**, we applied the communication inference to address the exploratory task of uncovering activities central to the functioning of the smart home service. The following matrix multiplication was used to obtain activities communicating to each other: $\text{USD } [Ac|E] \times \text{WGB } [E|Ac] = \text{WIB } [Ac|Ac]$ [12]. In Figure 5.6.a, we see the resulting matrix of the WIB [Ac|Ac] domino tile. The activities in the row (e.g., “visualize”) follow the activities in the column (e.g., “analyze_alert”). We exhibit an extended version of the domino tile WAW [Ac|Ag] in Figure 5.6.b to observe which devices are behind activities. The original WAW [Ac|Ag] had no activities associated with “Device1”, however “Sensor1” and “Sensor2” are delegates of “Device1” as stated by the AOB [Ag|Ag] domino tile. By combining $\text{WAW } [Ac|Ag] \times \text{AOB } [Ag|Ag] = [Ac|Ag]$ we obtain the indirect activities associated with “Device1”. Then, we add up the resulting domino tile to the original WAW [Ac|Ag] to produce the extended version in Figure 5.6.b. We can see that the only agents associated with more than one activity are “Device1” (two activities) and “Server” (three activities), highlighting the former as a responsible agent and suggesting the latter as a demanded agent.

Analyzing together both domino tiles of Figure 5.6, we realize that temperature (“Sen-

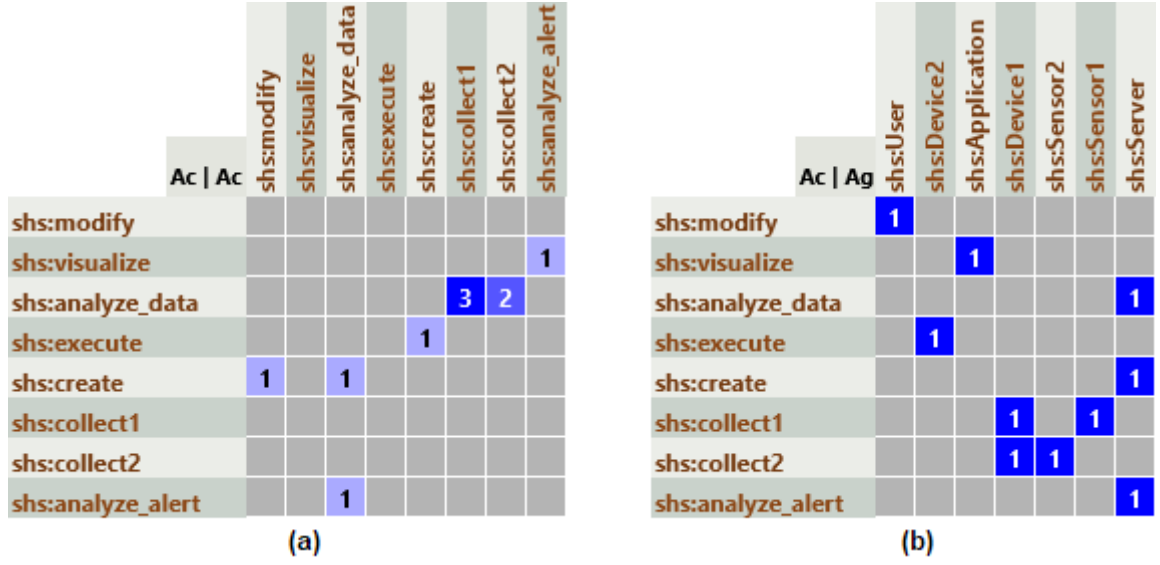


Figure 5.6: Activities communicating to each other (a); and Activities (rows) associated with agents (b).

sensor1”) and humidity (“Sensor2”) were collected (“collect1” and “collect2” activities) by the sensors (“Sensor1” and “Sensor2” agents), as we can see in Figure 5.6.b. Then the collected information was analyzed (“collect1” and “collect2” inform “analyze_data” in Figure 5.6.a) by the “Server” (“analyze_data” is associated with “Server” in Figure 5.6.b). After the analyses of the collected information from sensors, an alert analysis follows: “analyze_alert” was informed by “analyze_data” in Figure 5.6.a, also performed by the “Server” (in Figure 5.6.b, “analyze_alert” is associated with “Server”). Finally, the “visualize” activity (“visualize” was informed by “analyze_alert” in Figure 5.6.a) was executed by the “Application” agent (“Application” issues “visualize” in Figure 5.6.b), on the users’ smartphone. To summarize: the “Server” analyzed temperature and humidity collected by the sensors and decided to issue an alert that was visualized on the “Application” in the user’s smartphone.

Next, we analyze the activities centrality through the *eigenvector centrality* graph available in Prov-Dominoes for square matrices to measure the importance of the activities. The resulting graph is exhibited in Figure 5.7. The graph presents the activity “analyze_data” with the highest centrality score (stronger blue shade), suggesting it as the most important activity. Observing the graph together with the matrix of Figure 5.6.b, we can see the activities with higher score are issued by the *agent* “Server”, suggesting a dependence of an *agent* operating from the cloud.

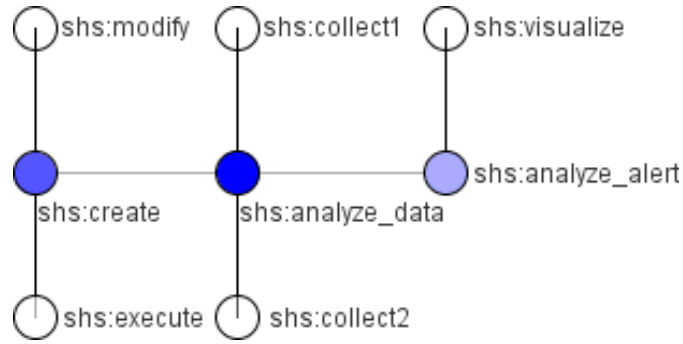


Figure 5.7: *Eigenvector centrality* graph on activities.

RQ1.1. *How Prov-Dominoes uncovers implicit information?*

In the provenance, there was no data direct relating activities (commands issued to devices) to each other. Such relation was made explicit by the communication inference: $USD [Ac|E] \times WGB [E|Ac] = WIB [Ac|Ac]$. After uncovering activity communications, the centrality analysis of activities became possible.

RQ1.2. *How Prov-Dominoes provides a holistic perspective?*

The centrality graph in Figure 5.7 along with the matrices of Figure 5.6 assisted in providing a wholesome understanding of what the provenance of the smart home represents. Altogether, the analysis indicated how the collections from the sensors subsided a decision from an agent operating from a cloud (“Server”) to issue an alert visualized on the user’s smartphone.

RQ1.3. *How Prov-Dominoes supports concise analysis?*

The original provenance graph of Figure 5.2 has around four dozen boxes. In contrast, the centrality graph in Figure 5.7 has only eight nodes, from which it is possible to observe the collections from sensors (“collect1” and “collect2”) and, together with the matrix of Figure 5.6.b, realize which agents are responsible for such activities, respectively: the sensors and the “Server”. Altogether, such concise analysis needs fewer elements to be observed, yet retaining central information to the provenance understanding.

5.1.3 Exploratory Practices

This section discusses transformations and visualizations that were articulated together during the explorations, working as agnostic **Exploratory Practices**. Moreover, we sum-

marize how such practices relate to the research questions of the effectiveness evaluation.

Exploring Combinations (EP1): We realized that exploring combinations involving provenance inferences, or just domino tiles targeted for analysis, allowed us to enrich domino tiles or derive new ones for all case studies by making implicit data explicit. As new data are uncovered, new exploration avenues unfold, contributing to a richer analysis. In particular, the following provenance inferences revealed themselves effective in unveiling implicit data:

- $\text{USD } [Ac|E] \times \text{WGB } [E|Ac] = \text{WIB } [Ac|Ac]$ [12] and
- $\text{WGB } [E|Ac] \times \text{USD } [Ac|E] = \text{WDF } [E|E]$ [15].

Centrality Analysis (EP2): Provenance graphs with many nodes and annotated elements such as the one in Figure 5.2 may overwhelm the user’s capability to visually extract knowledge. The centrality analysis provides a per-type approach to analyze provenance data by visualizing reduced graphs such as the centrality graphs of activities, agents, or entities. Moreover, such a concise perspective provides indications of importance on the nodes, as discussed in Section 3.3. For the **Smart Home case study**, we were able to identify important activities by employing centrality analysis.

Dependency Analysis (EP3): Dependency analysis is a useful technique that has many applications in software engineering and component-based systems (CBS) [2]. For the **Workflow case study**, where a series of components (modules) articulate together to perform some task, we employed the *transitive closure* operation to subsidize a dependency analysis. The analysis was capable of highlighting relevant parts of the workflow and providing an understanding of parameter influences.

Sparsity and Patterns Analysis (EP4): The matrix visualization itself provides an environment for sparsity and pattern analysis. The former can visually indicate the degree of matrix sparsity and provide an idea of data entropy. When done after sorting domino tiles, the latter may uncover patterns in rows and columns. Altogether, cell rearrangements and matrix visualization played an effective role in exploring the first two case studies, revealing patterns that assisted the analysis.

Dispersion Outlook (EP5): By applying the *Z-score* operation for the **first case study**, we realize how assistful it was in providing a holistic perspective of the domain, rather than a pure data dispersion view. The standard deviation and other transformations, such as binarization and filters, provided together effective ways to shape data for a holistic analysis.

Table 5.1: Exploratory Practices addressing research questions of the effectiveness evaluation.

Case Study	RQ1.1	RQ1.2	RQ1.3
First (Animals)	EP1, EP4	EP5, EP6	EP4
Second (Workflow)	EP1, EP4	EP3, EP6	EP4
Third (Smart Home)	EP1	EP2	EP2, EP4

Dimension Reduction (EP6): For the first two case studies, we realized that removing empty rows/columns with the *Trim* operation reduced the matrix and facilitated pattern identification among cells. However, it is important to interpret the empty rows/-columns before proceeding with such a reduction to avoid losses of relevant information. An empty row or column is a pattern itself and may carry relevant meaning.

We summarize in Table 5.1 the exploratory practices observed in each case study for each research questions of the effectiveness evaluation. We can see that **Exploring Combinations (EP1)** and **Sparsity and Patterns Analysis (EP4)** were employed in all case studies, to answer different research questions, suggesting them as effective exploratory practices for unveiling implicit data (**RQ1.1**) and supporting from concise analysis (**RQ1.3**), respectively.

5.2 Efficiency

In this section, we present the efficiency evaluation. For such analysis, we process thousands of provenance relations in both GPU and CPU to answer **RQ2**. First, we describe the materials and methods, and then we present and discuss the results.

5.2.1 Materials and Methods

The corpus of this evaluation consists of thousands of tweets (a post on Twitter) collected to measure how Prov-Dominoes performs on different processing modes and over increasing data volumes. We used a free subscription of the Twitter API to collect data, limiting the searches to seven days.

As the COVID-19 pandemic was a growing interest in social networks, we decided to collect tweets about Brazilian officials managing policies to mitigate the pandemic. The collected tweets subsided a sentiment analysis during the time frame from 05/20/2020 to 05/27/2020. We could collect more than four thousand tweets in this context.

When parsing the data to provenance, we identified the tweets as entities, the users as agents, and publishing the tweets as activities. We categorized the users according to their influential scope (their followers), by using the “prov:type” attribute and the hashtags used in a publishing were set as an *activity-type*. We used “::” as a separator to handle multiple hashtags:

```
activity(pub..., [prov:type = "#hashtag1::#hashtag2"])).
```

As result, Prov-Dominoes generated five domino tiles: WAW [Ac|Ag], WGB [E|Ac], WAT [E|Ag], *activity-type* [Ac|T] (hashtags), and *agent-type* [Ag|T] (number of followers).

The explorations focused on an exploratory task described in an EPS available in Appendix C. The exploratory task aimed at providing a perspective on the influential scope of users mentioning the Executive leaders. Initially, the EPS targeted only the full collection of 4,176 tweets both on CPU⁴ and GPU⁵. Further, we decided to split the original collection into eight subsets of tweets, grouped according to their orders of magnitude⁶ (m):

- Group 1 ($m = 2$): 100, 200, 300, and 500; and
- Group 2 ($m = 3$): 1000, 2000, 3000, and 4,176.

The reason for different collection sizes was to understand the curve behavior in different modes over increasingly data volume as well as different orders of magnitude. Then, we organized a drill where we ran the EPS targeting each subset upwardly. For instance, a drill executes the EPS targeting subset 100, then 200, then 300, until subset 4,176, then we collect the aggregated time of the drill (sum of the timings of the executions of all subsets). As we were interested in evaluating the processing, rather than the memory management, we run the drill five times, eliminating extreme values and computing the average of the remaining three in order to mitigate the memory management impact. For all drills, we ran Prov-Dominoes in “auto-load” mode, where Prov-Dominoes executes the EPS after booting. Moreover, we set tuning and telemetry parameters to “true”. The former avoids the optional matrix sorting before multiplication and the latter outputs execution times on the JVM (Java Virtual Machine) console.

⁴Intel Core i5 @ 2.90 GHz, 8 GB RAM.

⁵NVidia GeForce GT 1030, 2GB of dedicated RAM and 4GB of shared RAM.

⁶ $N = a \times 10^b$, where $1 \leq a < 10$ and b is the order of magnitude.

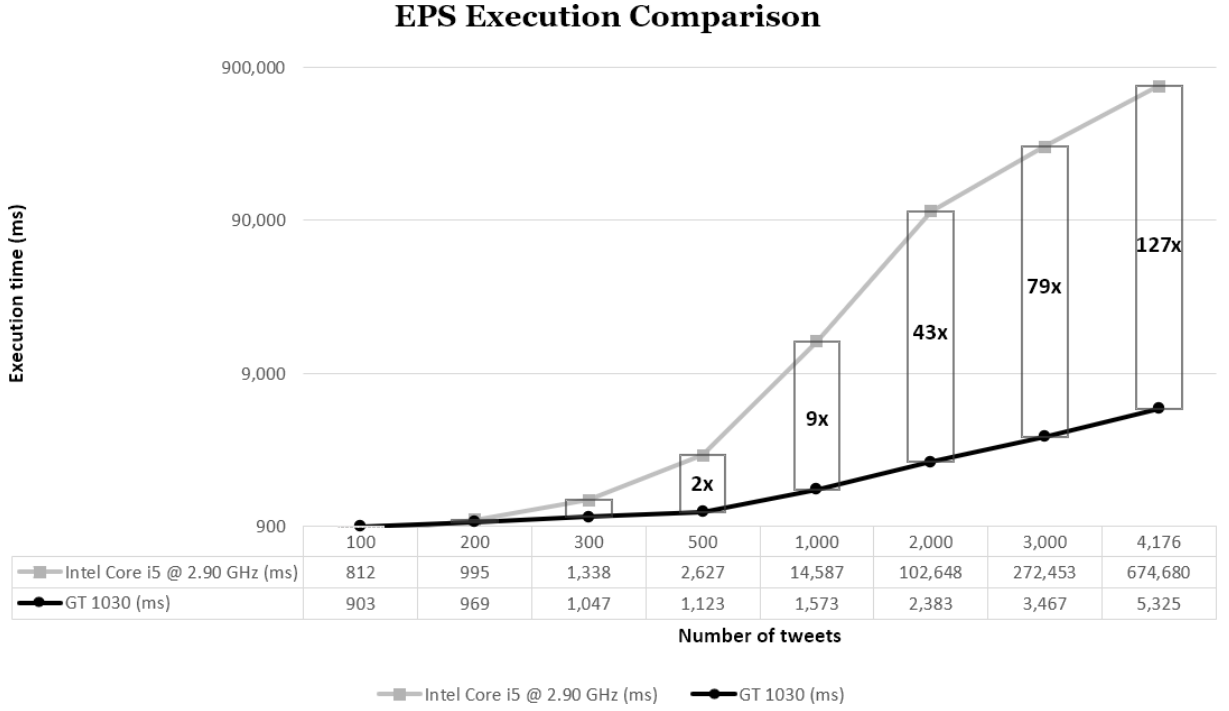


Figure 5.8: CPU-GPU Comparison with speedups.

5.2.2 Results and Discussion

The combined results of the drills on CPU and GPU are depicted in Figure 5.8. The execution times are in logarithmic scale. The numbers between the bars, rounded to the nearest integer, indicate the speedup between the processing modes, representing how many times GPU was faster than CPU. GPU was approximately 127 times faster for the largest tweet collection than CPU. CPU was marginally (less than 100 milliseconds) faster for 100 and 200 tweets. In this case, GPU was slower because memory transfer adds time to the GPU execution. Such overhead is a common bottleneck between integrated CPU-GPU applications. From 300 tweets on, GPU becomes faster as the combinations and transformations time in CPU starts to exceed the memory transfer overhead. For 500 tweets, GPU already performs more than two times faster. For the complete dataset of 4,176 tweets, Prov-Dominoes runs in 5 seconds in GPU, a quite reasonable time for real-time exploratory analyses when compared to 11 minutes in CPU. Such time difference clearly shows that the GPU processing feature of Prov-Dominoes was paramount for allowing interactive explorations over large datasets, previously almost impracticable in CPU.

Going further, we discuss the results within magnitude groups for the individual commands used in Prov-Dominoes. We observe in Figure 5.9 stacked bars representing executions times of the commands in *EPS 2* for GPU and CPU side by side. For instance,

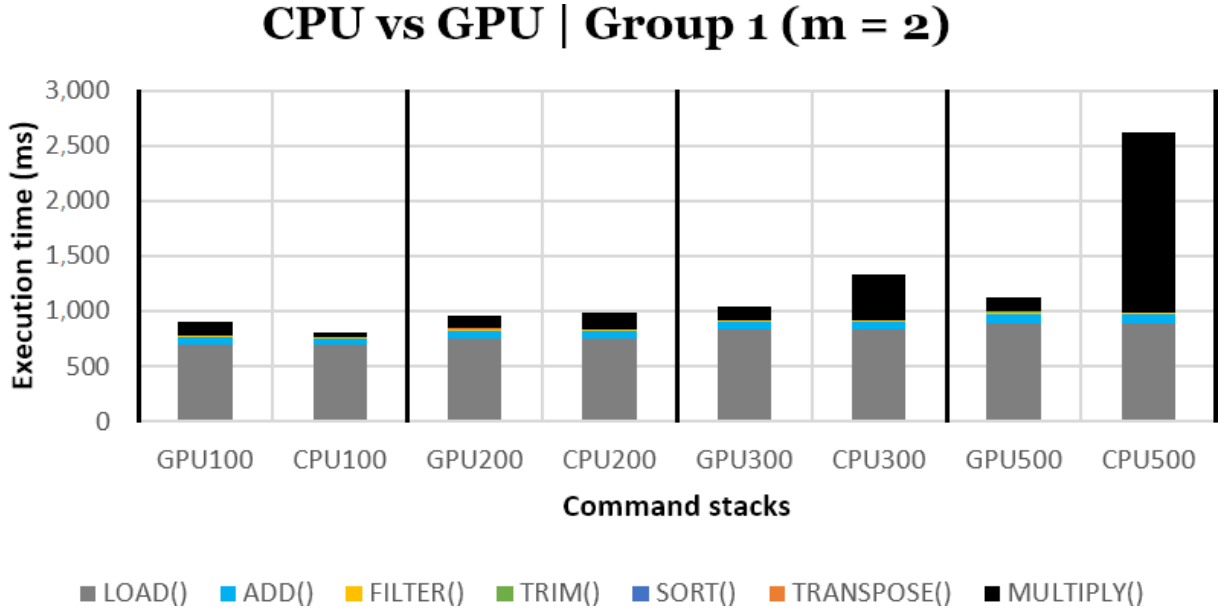


Figure 5.9: CPU-GPU Comparison (Group 1) on the commands time stack.

GPU100 and CPU100 mean *EPS 2* executions targeting 100 tweets for GPU and CPU, respectively. We can see the LOAD command has the same size in CPU and GPU for the same number of tweets. This happens because the LOAD command is always performed in CPU due to its dependence on I/O (file reading). The remaining commands are performed according to the processing mode. We can see that the multiply (top black bar) command takes more time as the number of tweets increases for CPU, up to the point of taking more than two times than the LOAD command for 500 tweets. Conversely, the multiply command remains more or less stable in GPU. Thus, for this order of magnitude (Group 1), we can observe that, in only one situation, a command surpasses the LOAD command in time: the multiply command running on CPU for 500 tweets.

Considering Group 2, depicted in Figure 5.10, we can see that, altogether, the GPU execution time remained more or less the same. Conversely, the multiply command in CPU escalated overwhelmingly over the other commands.

RQ2. *How efficient is Prov-Dominoes when running in GPU in comparison to CPU?*

In our evaluation, from 300 tweets on, GPU was faster. Even for less than 300 tweets, the difference in favor of CPU was only marginal (less than 100 milliseconds), suggesting that the GPU processing mode fits any data volume. For larger data volumes, the speedups increased considerably, up to the point of approximately 127 times faster for the largest data volume (4,176 tweets) in relation to the CPU.

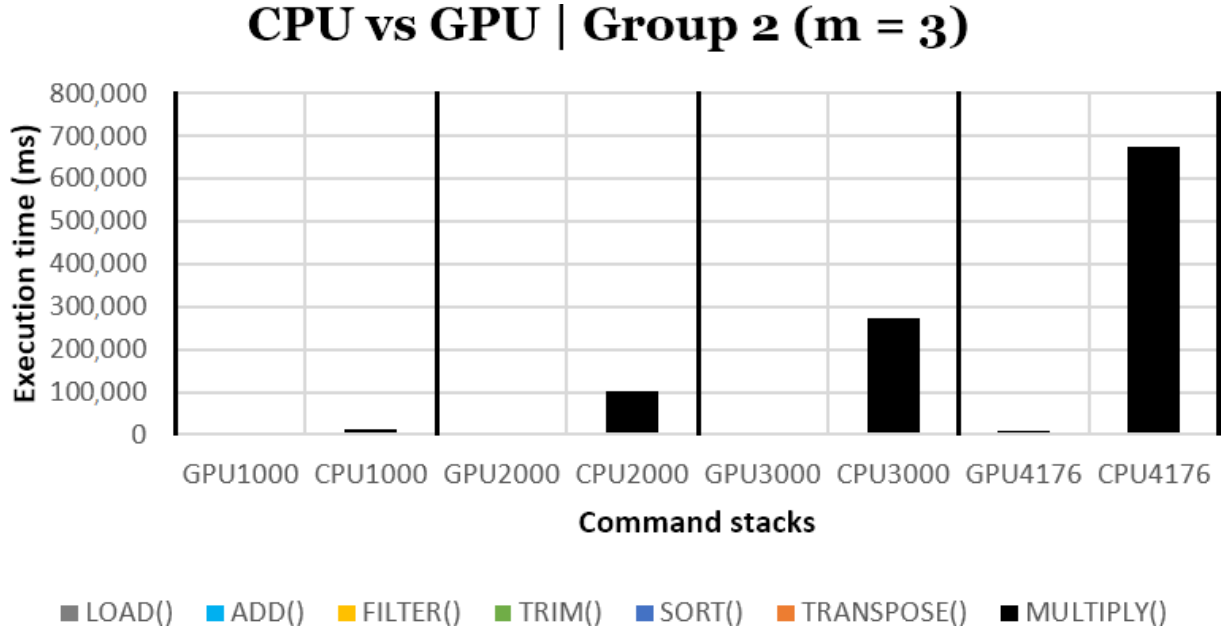


Figure 5.10: CPU-GPU Comparison (Group 2) on the commands time stack.

5.3 Threats to Validity

Besides our efforts to avoid limitations to our study, our approach and results are subject to some threats to validity, as discussed in the following.

Internal. Two internal threats stand out: (1) during the efficiency evaluation, we could have made the CPU implementations faster by implementing them in C or assembly. However, the results in favor of GPU were so compelling that we refrained from duplicate the Java implementations on faster languages; and (2) we only considered one opinion on the second case study (Workflow). Although this opinion may be biased, we tried to mitigate it by seeking for the opinion of the workflow’s author. Finally, the results from the Animals and Smart Home case studies were not validated by specialists. However, they were selected due to their familiar nature, enabling readers to interpret the results themselves.

Construct. During the effectiveness evaluation, we did not always stick to concrete measures. While addressing the research question **RQ1.2** (*How Prov-Dominoes provides a holistic perspective?*), we did it in an interpretive way, relying on an abstract notion of the meaning of holistic. However, we tried to mitigate such a threat to **RQ1** by considering concrete measures for the remaining two sub-questions by identifying explicit data for **RQ1.1** and patterns and graph comparisons in **RQ1.3**.

External. We did not account for how the tool would be operated by other users,

rather than the authors. New users may experience steep learning curves while exploring with the tool. The exploratory practices identified after our explorations on the effectiveness evaluation attempt to mitigate such learning curves by suggesting practices that may guide their explorations. Conversely, the exploratory practices may not be as assistful as they were due to the reduced number of considered case studies. Although we used only three case studies for the effectiveness assessment, we tried to mitigate such a threat by considering distinct domains.

Conclusion. The memory transfer bottleneck discussed in Section 5.2.2 is a sensitive matter while measuring the timing of the executions. Such sensitiveness happens because memory management has different architectural organizations between CPU and GPU, leading to different management strategies. When performing the same operation repeatedly and taking the average time, the results may vary significantly. The memory management impact can contribute to either decrease or increase the execution time.

Chapter 6

Related Work

Over the years, provenance’s research and the number of available tools has grown, the community’s knowledge of the many factors and goals relevant for effective provenance support has also broadened. However, the variety of perspectives can make it challenging to assess the specific aspects and purposes of provenance that are targeted by any particular project [51].

Depending on the research area, provenance interpretation can vary. For example, the design of computational tasks for scientific experiments often regard provenance as the history of computational workflow [31], while other interpretations focus on the history of gameplay states [37]. According to Ragan et al. [51], different perspectives and applications of concepts become problematic for interpreting and coordinating outcomes from different provenance projects, for communicating ideas within the visualization community, and for allowing new-comers to clearly understand the research space.

In order to clear the research space and contextualize our related work, we considered the following three topics as scope: Provenance Visualization Analysis (Section 6.1), Provenance Data Analysis (Section 6.2), and Provenance and GPU (Section 6.3).

The topics were organized in this manner, as they share issues that contributed directly or indirectly to the research that culminated in Prov-Dominoes, the tool designed as result of this dissertation.

6.1 Provenance Visualization Analysis

Iliinsky and Steele [35] identify two categories of data visualization: exploration and explanation. Exploratory visualization is designed for a researcher who is not certain what

is in the data or what they are looking for, typically when dealing with large volumes of data. Explanatory visualization, on the other hand, is a visualization that takes place when a researcher knows what the data has to say, and is trying to tell that story to someone else. The two serve different purposes, and there are tools and approaches that may be appropriate for one and not the other [10].

Among the exploratory visualization tools, we list two domain-specific tools: GENI [10] (Visualization of network data provenance) and InProv [7] (visualizations of file system provenance data). As domain-specific tools, they are prepared to deal with large volumes of data and, individually, provide some interesting features, such as interactive radial-based tree layout, abstract views, breadth-first searches, and graph comparison. However, as visualization centered tools, they do not provide data combination. Moreover, these features are not available in an integrated way, hindering the analysis due to visualization and manipulation restrictions.

In comparison, Prov-Dominoes is able to represent and provide visualization analyses for provenances from different domains, as long as they are defined or exported to PROV-N or PROV-XML formats. Moreover, Prov-Dominoes support interactively combination of data, abstracted as domino tiles, which open avenue for further analyses and visualizations. Finally, by enabling to import and export the explorations performed in the tool, it reduces the complexity to integrate and collaborate with different analyses while exploring provenances.

As explanatory visualization tools, we can cite some general-purpose visualization tools, compatible with W3C PROV, such as ProvToolbox [42], Provenance Explorer [18], and Prov Viewer [38]. ProvToolbox was one of the first W3C PROV compatible tools, converting the PROV-DM representations into various formats. However, it lacks a built-in visualization and requires a generic graph tool to visualize the provenance data. Provenance Explorer takes RDF-based provenance outputs from capture systems and dynamically generates customized views of provenance trail. Prov Viewer enables users to explore provenance data over graphs interactively. Different from Provenance Explorer, Prov Viewer provides data combinations through graph merges. Despite being graph-based, neither tools provide score-based centrality analysis as Prov-Dominoes' *eigenvector centrality*. Despite their visualization features, such tools are sensitive to large data, overwhelming the user's ability to extract knowledge in a visual way. On the other hand, Prov-Dominoes' matrix visualization can represent hundreds or thousands of graph nodes coupled with guiding features, such as row/column patterns unveiled by matrix sorting

and data dispersion through *Z-Score* operation.

Two *Workflow Management Systems* (WfMSs) are widely known for supporting scientific experiments: Taverna [34] and VisTrails [9]. Taverna is an application that eases the use and integration of the growing number of molecular biology tools and databases available on the web. It allows bioinformaticians to construct workflows or pipelines of services to perform various analyses, such as sequence analysis and genome annotation. VisTrails allows users to navigate through workflow versions intuitively, visually comparing different workflows and their results while examining the actions that led to a result. Different from Taverna, VisTrails provides provenance of workflow evolution, and has built-in provenance capture of workflow executions. Although Taverna needs plugins for provenance capture of workflow executions, such as Taverna-PROV¹, the WfMS is continually evolving and improving workflow analysis features after transitioning to Apache Incubator². Despite their concise visualization through workflows, they hide parameters used by modules in the workflow view and lack features to analyze module parameters' influences, needing to delegate such analysis to other provenance tools.

6.2 Provenance Data Analysis

In this section, we discuss three exploratory data analysis tools related to provenance and how such tools compare to or influenced our work. The first is a software package within the statistical programming environment R to assist exploratory analysis on provenance data related to geological sediments. The second is an approach that integrates provenance from Python scripts with IPython notebooks to support interactive and exploratory analyzes. The third is Dominoes [23], a tool that organizes provenance data from Git repositories into multiple matrices that can be treated as domino tiles for further data combinations and analysis. In all cases, we consider provenance as data about the history of objects (e.g., data about the history of geological sediments) or digital objects (e.g., data about the evolution or history of a git repository).

When analyzing datasets on geological sediments, the analyst needs to apply, combine and interpret different statistical resources. Different levels of statistical complexity arise when multiple samples are compared to each other, or when multiple provenance proxies are applied to multiple samples [55]. Pieter Vermeesch developed a provenance package³

¹<https://github.com/taverna/taverna-prov>

²Apache Incubator provides services to projects which want to enter the Apache Software Foundation.

³<https://cran.r-project.org/web/packages/provenance/index.html>

within the statistical programming environment R to enable geologists to interactively explore the provenance data of geological sediments. The package provides means to subsidize *Multidimensional Scaling* (MDS) and *Principal Component Analysis* (PCA). Moreover, provenance data can be augmented with compositional information as biplots easing comparison among multiple datasets. The idea of separating data for further compositions and analysis is also a key feature in Prov-Dominoes. For instance, combinations between domino tiles and per-type analysis as in the eigenvector centrality of agents, activities or entities.

Among scientific experiments, noWorkflow [46] emerged is an open-source tool that systematically and transparently collects provenance from Python scripts, including data about the script execution and how the script evolves over time. When used on a script of scientific experiments, the tool enables scientists to analyze multiple trials, compare them, and understand their history. In order to enable scientists to explore on the provenance collected by noWorkflow, a command line option on noWorkflow allows the generation of a notebook file with the code used for loading the trial. Then, a scientist can perform analysis on any data collected by noWorkflow through SQL queries, Prolog queries, object properties, and graph visualization [49]. Additionally, it is possible to get provenance data from different queries, and combine them with custom Python code.

Another example of data analysis tool is Dominoes. The tool organizes data extracted from Git repositories into multiple matrices that can be treated as domino tiles (e.g., [commit|method]). It allows connecting such domino tiles based on a set of matrix operations to derive additional ones. Although Dominoes is a tool aimed at Git repositories, these repositories can be exposed as provenance, in the way demonstrated by Git2PROV [25]. The characterization of Git repositories as provenance gave birth to the idea of a tool similar to Dominoes, now focusing on an agnostic provenance exploratory tool. For its influence to Prov-Dominoes as a provenance data analysis tool, we mention Dominoes in this section.

Both Dominoes and noWorkflow provide data combination over the captured provenance. The former provides various connecting possibilities among the extracted data, and the latter provides combination in the form of merged graphs for diff-based analysis of trials. However, noWorkflow's diff-based analysis is restricted to two trials. If their trials were exported to PROV-N, it would be possible to aggregate or combine multiple trials in Prov-Dominoes. As a fork of Dominoes, Prov-Dominoes adds features such as: *transitive closure*, *eigenvector centrality*, filters and sorting transformations. Coupled together,

these features unveil further analysis possibilities.

Although useful, Vermeesch’s Provenance R Package, noWorkflow and Dominoes are tied to the specifics of the objects or environments they target, respectively: R statistical programming environment, Python scripts and Git repositories. Prov-Dominoes support to a general provenance representation format like PROV-N makes it capable to represent distinct provenance domains.

6.3 Provenance and GPU

GPUs are getting popularly utilized for multi-purpose applications in order to enhance highly performed parallelism of computation [36]. In this section, we present some provenance related works taking benefits of GPUs.

Purawat et al [50] propose an automated workflow tool to perform molecular dynamics simulations that capitalizes on the capabilities of the Kepler platform to deliver a flexible, intuitive, and user-friendly environment. They use AMBER GPU code for a robust and high-performance simulation engine. Additionally, the workflow tool reduces user input time by automating repetitive processes and facilitates access to GPU clusters, whose high-performance processing power makes simulations of large numerical scale possible. The workflow tool also performs systematic analysis on the simulation outputs and enhances simulation reproducibility.

Bruder et al. [8] present an approach for the visualization and interactive analysis of dynamic graphs that contain a large number of time steps. Focus is put on the support of analyzing temporal aspects in the data and dynamic graphs based on the concept of space-time cubes. The implementation is GPU-accelerated, enabling interactive exploration on large data sets. According to their work, four classes of approach are key to the analysis of large and complex graph data: data views, aggregation and filtering, comparison, and evolution provenance. Implementations of the respective methods are presented in an integrated application, enabling interactive exploration and analysis of large graphs.

In the previous section, we presented Dominoes as a tool supporting exploratory data analysis. Yet, Dominoes also has GPU processing capabilities [24]. The domino-matrix representation introduced by Dominoes allows for fast and efficient processing of a large volume of data by using a highly parallel architecture, such as GPUs. Inspired by Dominoes, Prov-Dominoes extended the set of GPU implementations existent in Dominoes by adding new operations, such as: addition, subtraction, transitive closure and binarization.

Chapter 7

Conclusion

Although provenance graph visualization is common, such support is limited when it is necessary to combine data to uncover implicit information. Moreover, in complex domains, the provenance graphs can be composed of thousands of vertices and edges, becoming visually cluttered and limiting the user's ability to visually and interactively analyze the data. Prov-Dominoes addresses such challenges by providing concise visualizations through matrices and reduced centrality graphs, besides offering features and practices to combine, transform, and reduce data. Such features allow unveiling implicit information and contributing to a more comprehensive perspective of the data domain. In Section 7.1 we summarize the contributions and in Sections 7.2 and 7.3 we present the limitations and future work, respectively.

7.1 Contributions

This work introduced an approach for exploratory analysis of provenance data. By interactively combining provenance data in the form of domino tiles, the analyst can explore provenance on different perspectives (e.g., matrix visualization and eigenvector centrality graph) and PROV-DM Concepts (types and relations). The provenance data representation as matrices is also a contribution, acting as a link between data analysis and visualization analysis of provenance data. Moreover, a Java tool was developed to make the proposed approach viable: Prov-Dominoes. Our tool is compatible with PROV-N notation, allowing its adoption in different domains and applications. Furthermore, it can enrich provenance data with few explicit expressions, applying provenance inferences to uncover new analysis usually neglected due to rich information buried in the data.

We presented three case studies from different domains, illustrating the agnostic po-

tential of the analyses performed in Prov-Dominoes. We uncovered implicit information such as influencing parameters in a workflow execution and central activities in a smart home service. Moreover, our analyses provided wholesome perspectives of the case studies, such as animals' ruling activities, and the main components of a workflow.

As a guideline for provenance exploratory analysis, this work proposed a set of practices to assist users while performing provenance explorations with Prov-Dominoes. The practices indicate how to articulate together transformations and visualizations available in the tool to extract information from the provenance and unveil more analysis possibilities. As the practices showed themselves effective over explorations on three distinct domains, we consider them agnostic and potentially useful for other domains.

Finally, we employ the use of *High-Performance Computing* (HPC) by taking advantage of GPU during provenance explorations. As the underlying abstraction of the domino tiles were matrices, it made possible the use and implementation of matrix transformations in GPU to empower processing. The efficiency evaluation in the context of Twitter sentiment analysis revealed that GPU was up to 127 times faster for combining thousands of relations when compared to CPU.

7.2 Limitations

When used in CPU mode only, the tool does not accommodate large provenance data, as the matrix processing time can be extremely long. In such situations, we strongly recommend to use the tool in a computer with GPU. As showed during efficiency evaluation, low cost GPUs such as the NVIDIA GeForce GT 1030 already can produce significant performance results on explorations over large provenance data.

We did not perform usability assessment of Prov-Dominoes, being unknown the tool learning curve when used by different users. Although similar to Dominoes, which provides satisfactory user studies [23], such studies are restricted to one domain (git repositories) and its associated stakeholders. As Prov-Dominoes is fit for other domain uses, it may be associated with steep learning curves, depending on the user scenario.

Due to the limited number of case studies considered, the proposed exploratory practices, although agnostic, may not reveal themselves as useful as expected. Moreover, there may be other practices not yet identified and that may prove useful according to the domain.

7.3 Future Work

Further studies could be made aiming at incorporate other matrix operations that support exploratory analysis in the context of provenance. For example, investigate the use of inverse matrix in the context of provenance matrices. Additionally, implementing more algorithms for clustering and sorting, and extending to GPU current matrix transformations only available in CPU, such as *Trim* and *Word on Row/Column*.

Detailed comparison studies could be made between the provenance matrix representation introduced by this work and provenance graphs. Focusing on establishing diverse exploratory tasks, domains, and users to better understand each approach's pros and cons.

Investigating the tool usage in other domains and focusing on how users explore with the tool may enrich the current exploratory practices and give insights on how improve Prov-Dominoes features to better assist provenance exploration. For instance, collecting various EPS from different users/domains and study what they have in common. These results could indicate additional practices or suggest new features.

Another possibility would be to study whether the identified exploratory practices could be considered a canonical exploratory resource. For instance, regardless of the approach to provenance analysis, is it possible to answer the effectiveness evaluation's research questions without employing the exploratory practices?

Inspired by the findings resulting from the explorations in the Animals case study, it would be possible to further examine the adoption of such explorations in education domains. This would promote interactive analysis and information discovery, particularly in contexts with different knowledge levels, such as school grades.

In order to make the tool accessible to more domains, we developed, as mentioned in Section 4.1, an API to convert application logs into PROV-N files. However, we did not further investigate which insights the provenance of such logs could provide. We suspect converting logs into provenance and analyzing them in the tool may reveal relevant information and assisting activities such as log auditing and change decisions in the tasks that generated the logs.

Working on integrating Prov-Dominoes to existing provenance tools can prove to be a worthwhile effort: providing enrichment of analysis and interoperation capabilities among different provenance ecosystems. For instance, Prov Viewer and Prov-Dominoes could be

merged into a provenance suite for provenance visualization and analysis.

Finally, as the tool provides means to provenance inferences, it could be adapted to work as an environment for producing normal forms of PROV instances, and assist in checking PROV validity and equivalence properties [16]. For instance, we can cite the provenance generated in the context of independent scientific experiments. Two similar experiments from independent projects may have similar provenances. In order to better collaborate, it is important to confirm if such provenances are formally equivalent.

References

- [1] ACKERMAN, M. J. The visible human project. *Proceedings of the IEEE* 86, 3 (1998).
- [2] AMALARETHINAM, G.; MAITHEEN, P.; HAMEED, S. Dependency analysis for component based systems using graphs. *International Journal of Applied Engineering Research* 10 (12 2015).
- [3] BASTIAN, M.; HEYMANN, S.; JACOMY, M., ET AL. Gephi: an open source software for exploring and manipulating networks. *Icwsm* 8, 2009 (2009), 361–362.
- [4] BELHAJJAME, K.; B’FAR, R., ET AL. Prov-DM: The PROV Data Model. Tech. rep., W3C, 2013.
- [5] BERNERS-LEE, T.; JAFFE, J., ET AL. World Wide Web Consortium (W3C), 1994.
- [6] BONACICH, P. Power and centrality: A family of measures. *American journal of sociology* 92, 5 (1987), 1170–1182.
- [7] BORKIN, M. A.; YEH, C. S.; BOYD, M.; MACKO, P.; GAJOS, K. Z.; SELTZER, M.; PFISTER, H. Evaluation of filesystem provenance visualization tools. *IEEE Transactions on Visualization and Computer Graphics* 19, 12 (2013), 2476–2485.
- [8] BRUDER, V.; LAHMAR, H. B.; HLAWATSCH, M.; FREY, S.; BURCH, M.; WEISKOPF, D.; HERSCHEL, M.; ERTL, T. Volume-based large dynamic graph analysis supported by evolution provenance. *Multimedia Tools and Applications* 78, 23 (2019), 32939–32965.
- [9] CALLAHAN, S. P.; FREIRE, J.; SANTOS, E.; SCHEIDEGGER, C. E.; SILVA, C. T.; VO, H. T. Vistrails: visualization meets data management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (2006), pp. 745–747.
- [10] CHEN, P.; PLALE, B.; CHEAH, Y.-W.; GHOSHAL, D.; JENSEN, S.; LUO, Y. Visualization of network data provenance. In *2012 19th International Conference on High Performance Computing* (2012), IEEE, pp. 1–9.
- [11] CHENEY, J.; MISSIER, P.; MOREAU, L. Attribution Inference. Tech. rep., W3C, 2013.
- [12] CHENEY, J.; MISSIER, P.; MOREAU, L. Communication Inference. Tech. rep., W3C, 2013.
- [13] CHENEY, J.; MISSIER, P.; MOREAU, L. Constraints of the PROV Data Model. Tech. rep., W3C, 2013.

- [14] CHENEY, J.; MISSIER, P.; MOREAU, L. Delegation Inference. Tech. rep., W3C, 2013.
- [15] CHENEY, J.; MISSIER, P.; MOREAU, L. Derivation Inference. Tech. rep., W3C, 2013.
- [16] CHENEY, J.; MISSIER, P.; MOREAU, L. Normalization, validity, and equivalence. Tech. rep., W3C, 2013.
- [17] CHENEY, J.; MISSIER, P.; MOREAU, L.; SOILAND-REYES, S. PROV-N: The provenance notation. Tech. rep., W3C, 2013.
- [18] CHEUNG, K.; HUNTER, J. Provenance explorer—customized provenance views using semantic inferencing. In *International Semantic Web Conference* (2006), Springer, pp. 215–227.
- [19] CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *IEEE Transactions on software engineering* 20, 6 (1994), 476–493.
- [20] CLOSA, G.; MASÓ, J.; PROSS, B.; PONS, X. W3c prov to describe provenance at the dataset, feature and attribute levels in a distributed environment. *Computers, Environment and Urban Systems* 64 (2017), 103–117.
- [21] COMMITTEE, P. E., ET AL. Premis data dictionary for preservation metadata. *Preservation* (2008).
- [22] CRUCITTI, P.; LATORA, V.; PORTA, S. Centrality in networks of urban streets, chaos. *Quarterly of the American Institute of Physics* 16, 1 (2006).
- [23] DA SILVA JUNIOR, J. R.; CAMPAGNA, D. P.; CLUA, E.; SARMA, A.; MURTA, L. G. P. Dominoes: An interactive exploratory data analysis tool for software relationships. *IEEE Transactions on Software Engineering* (2020).
- [24] DA SILVA JUNIOR, J. R.; CLUA, E.; MURTA, L.; SARMA, A. Exploratory data analysis of software repositories via gpu processing. In *The International Conference on Software Engineering and Knowledge Engineering (SEKE)(Vancouver, Canada, 2014)* (2014), pp. 495–500.
- [25] DE NIES, T.; MAGLIACANE, S.; VERBORGH, R.; COPPENS, S.; GROTH, P. T.; MANNENS, E.; VAN DE WALLE, R. Git2prov: Exposing version control system content as w3c prov. In *International Semantic Web Conference (Posters & Demos)* (2013), pp. 125–128.
- [26] DEL RIO, N.; DA SILVA, P. P. Probe-it! visualization support for provenance. In *International Symposium on Visual Computing* (2007), Springer, pp. 732–741.
- [27] DUA, D.; GRAFF, C. UCI machine learning repository, 2017.
- [28] ELLSON, J.; GANSNER, E. R.; KOUTSOFIOS, E.; NORTH, S. C.; WOODHULL, G. Graphviz and dynagraph—static and dynamic graph drawing tools. In *Graph drawing software*. Springer, 2004, pp. 127–148.

- [29] FATAHALIAN, K.; SUGERMAN, J.; HANRAHAN, P. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (2004), ACM, pp. 133–137.
- [30] FORSYTH, R. Zoo dataset, uci machine learning repository, 2016.
- [31] FREIRE, J.; KOOP, D.; SANTOS, E.; SILVA, C. T. Provenance for computational tasks: A survey. *Computing in Science & Engineering* 10, 3 (2008), 11–21.
- [32] GODSIL, C.; ROYLE, G. F. *Algebraic graph theory*, vol. 207. Springer Science & Business Media, 2013.
- [33] GROTH, P.; MOREAU, L.; MISSIER, P.; FREIRE, J., ET AL. Ipaw, 2006.
- [34] HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; GOBLE, C.; POCOCK, M. R.; LI, P.; OINN, T. Taverna: a tool for building and running workflows of services. *Nucleic acids research* 34, suppl_2 (2006), W729–W732.
- [35] ILIINSKY, N.; STEELE, J. Classifications of visualizations. In *Designing Data Visualizations, Intentional Communication from Data to Display*. O'Reilly Media Sebastopol, CA, 2011.
- [36] KIM, S.; OH, J.; KIM, Y. Data provenance for experiment management of scientific applications on gpu. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)* (2019), IEEE, pp. 1–4.
- [37] KOHWALTER, T.; CLUA, E.; MURTA, L. Provenance in games. *XI SBGames* (2012), 162–171.
- [38] KOHWALTER, T.; OLIVEIRA, T.; FREIRE, J.; CLUA, E.; MURTA, L. Prov viewer: A graph-based visualization tool for interactive exploration of provenance data. In *International Provenance and Annotation Workshop* (2016), Springer, pp. 71–82.
- [39] MACKO, P.; SELTZER, M. Provenance map orbiter: Interactive exploration of large provenance graphs. In *TaPP* (2011), pp. 1–6.
- [40] MISSIER, P.; MOREAU, L. PROV-Overview. An overview of the PROV family of documents. Tech. rep., W3C, 2013.
- [41] MISSIER, P.; MOREAU, L., ET AL. Prov-dm relations at a glance, 2013.
- [42] MOREAU, L. Provtoolbox - java library to create and convert w3c prov data model representations. [Online]. Available: <http://lucmoreau.github.io/ProvToolbox/>, April 2016.
- [43] MOREAU, L.; CLIFFORD, B.; FREIRE, J.; FUTRELLE, J.; GIL, Y.; GROTH, P.; KWASNIKOWSKA, N.; MILES, S.; MISSIER, P.; MYERS, J., ET AL. The open provenance model core specification (v1. 1). *Future generation computer systems* 27, 6 (2011), 743–756.
- [44] MOREAU, L.; FREIRE, J.; FUTRELLE, J.; MCGRATH, R. E.; MYERS, J.; PAULSON, P. The open provenance model: An overview. In *International Provenance and Annotation Workshop* (2008), Springer, pp. 323–326.

- [45] MOREAU, L.; LUDÄSCHER, B.; ALTINTAS, I.; BARGA, R. S.; BOWERS, S.; CALLAHAN, S.; CHIN JR, G.; CLIFFORD, B.; COHEN, S.; COHEN-BOULAKIA, S., ET AL. The first provenance challenge. *Concurrency and computation: practice and experience* 20, 5 (2008), 409–418.
- [46] MURTA, L.; BRAGANHOLO, V.; CHIRIGATI, F.; KOOP, D.; FREIRE, J. noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop* (2014), Springer, pp. 71–83.
- [47] NEWBURY, D.; LIPPINCOTT, L. Provenance in 2050. In *Collecting and Provenance: A Multidisciplinary Approach*, J. Milosch and N. Pearce, Eds., 1 ed. Rowman & Littlefield Publishers, 2019, ch. 7, p. 104.
- [48] NEWMAN, M. E. The mathematics of networks. *The new palgrave encyclopedia of economics* 2, 2008 (2008), 1–12.
- [49] PIMENTEL, J. F. N.; BRAGANHOLO, V.; MURTA, L.; FREIRE, J. Collecting and analyzing provenance on interactive notebooks: when ipython meets noworkflow. In *7th {USENIX} Workshop on the Theory and Practice of Provenance (TaPP 15)* (2015).
- [50] PURAWAT, S.; IEONG, P. U.; MALMSTROM, R. D.; CHAN, G. J.; YEUNG, A. K.; WALKER, R. C.; ALTINTAS, I.; AMARO, R. E. A kepler workflow tool for reproducible amber gpu molecular dynamics. *Biophysical journal* 112, 12 (2017), 2469–2474.
- [51] RAGAN, E. D.; ENDERT, A.; SANYAL, J.; CHEN, J. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics* 22, 1 (2015), 31–40.
- [52] SCHROEDER, W.; MARTIN, K.; LORENSEN, B. *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Kitware, 2006.
- [53] SILVA, C. T.; FREIRE, J.; CALLAHAN, S. P. Provenance for visualizations: Reproducibility and beyond. *Computing in Science & Engineering* 9, 5 (2007), 82–89.
- [54] SKIENA, S. S. Transitive closure and reduction. In *The Algorithm Design Manual*, 2 ed. Springer, 2011, pp. 495–497.
- [55] VERMEESCH, P. Exploratory analysis of provenance data using r and the provenance package. *Minerals* 9, 3 (2019), 193.
- [56] WU, H.-M.; TZENG, S.; CHEN, C.-H. Matrix visualization. In *Handbook of data visualization*. Springer, 2008, pp. 681–708.
- [57] YADAV, A.; KHAN, R. Does coupling really affect complexity? In *2010 International Conference on Computer and Communication Technology (ICCT)* (2010), IEEE, pp. 583–588.
- [58] ZAKI, M. J.; MEIRA JR, W.; MEIRA, W. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

APPENDIX A – PROV-N from the guiding example

In the listing below, we show the complete PROV-N from our guiding example:

```
document
  agent(John)
  agent(Bill)
  entity(Water, [prov:type="Liquid"])
  entity(Barley, [prov:type="Grain"])
  entity(Wort, [prov:type="Liquid"])
  entity(Hopped_Wort, [prov:type="Liquid"])
  entity(Hops)
  entity(Yeast)
  entity(Priming_Sugars)
  entity(Beer)
  activity(Mashing)
  activity(Boiling)
  activity(Fermentation)
  used(Mashing, Water, -)
  used(Mashing, Barley, -)
  used(Boiling, Wort, -)
  used(Boiling, Hops, -)
  used(Fermentation, Yeast, -)
  used(Fermentation, Hopped_Wort, -)
  used(Fermentation, Priming_Sugars, -)
  wasGeneratedBy(Wort, Mashing, -)
  wasGeneratedBy(Hopped_Wort, Boiling, -)
  wasGeneratedBy(Beer, Fermentation, -)
  wasAssociatedWith(Mashing, John, -)
  wasAssociatedWith(Boiling, Bill, -)
  wasAssociatedWith(Fermentation, Bill, -)
  actedOnBehalfOf(John, Bill)
endDocument
```

APPENDIX B - List of all EPS commands

In the table below, we show all available EPS commands:

Command	Description
UNDO	Undo a command.
REDO	Redo a command.
MOVE	Move a tile on the Canvas to a (x,y) position.
SAVE	Save a tile on the Domino Tiles List.
LOAD_MATRIX	Load a .matrix file.
LOAD	Load a PROV-N or EPS file.
TRANPOSE	Transpose a domino tile.
MULTIPLY	Combine tiles into one by multiplying the underlying matrices.
SUM	Combine tiles into one by summing the underlying matrices.
SUBTRACT	Combine tiles into one by subtracting the underlying matrices.
ADD	Add a tile from Domino Tiles List to Canvas.
REMOVE	Remove a tile from Canvas.
AGG_ROWS	Aggregate the rows into one row (sum of all rows).
AGG_COLUMNS	Aggregate the columns into one column (sum of all columns).
CONFIDENCE	Generate confidence numbers related to the diagonals.
ZSCORE	Generate standard deviations from column average.
TRANSITIVE_CLOSURE	Set $1/n$ values on the cells, where n is row-column distances.
BINARIZE	Set 1 if cell is greater or equal to 1 and set 0 otherwise.
INVERT	Binarize cells and then zeros and ones are reversed.
SORT_ROW	Sort rows in ascending order.
SORT_COLUMN	Sort cells in ascending order.
SORT_ROW_GROUP	Sort rows by grouping rows.
SORT_COLUMN_GROUP	Sort cells by grouping columns.
DIAGONALIZE	Filter diagonal cells.
UPPER_TRIANGULAR	Filter diagonal and upper diagonal cells.
LOWER_TRIANGULAR	Filter diagonal and lower diagonal cells.
HPF	Filter cells equal or higher than a cutoff value.
LPF	Filter cells equal or lower than a cutoff value.
ROW_TEXT	Filter row cells matching to some word or regular expression.
COLUMN_TEXT	Filter column cells matching some word or regular expression.
TRIM	Eliminate empty rows and columns.

APPENDIX C - Efficiency Evaluation EPS

In the listing below, we show the EPS used for the efficiency evaluation. The EPS refers to the file “twitter-governadores_en.provn”, available at: <https://bit.ly/3ffn1ve>.

```
LOAD("twitter-governadores_en.provn")
p1 = ADD(WGB)
p2 = ADD(WAW)
p3 = MULTIPLY(p1, p2)
p4 = ADD(WAT)
TRANSPOSE(p4)
p5 = MULTIPLY(p4, p3)
p6 = ADD(AgT)
p7 = MULTIPLY(p5, p6)
TRANSPOSE(p7)
COLUMN_TEXT(p7, true, true, "jairbolsonaro|wilsonwitzel|jdoriajr|
    flaviodino|camilosantanace|romeuzema")
TRIM(p7)
SORT_COLUMN_GROUP(p7)
```