Observed vs. Possible Provenance
Research Track

Tom Blount
University of Southampton

Adriane Chapman
University of Southampton

Michael Johnson
Max Planck Institute for Radio Astronomy

Bertram Ludäscher
University of Illinois Urbana-Champaign

Abstract

Provenance has been of interest to the Computer Science community for nearly two decades, with proposed uses ranging from data authentication, to security auditing, to ensuring trust in decision making processes. However, despite its enthusiastic uptake in the academic community, its adoption elsewhere is often hindered by the cost of implementation. In this paper we seek to alleviate some of these factors, and propose the idea of *possible provenance* in which we relax the constraint that provenance must be directly observed. We categorise some existing approaches to gathering provenance and compare the costs and benefits of each, and illustrate one method for generating possible provenance in more detail with a simple example: inferring the possible provenance of a game of Connect Four. We then go on to discuss some of the benefits and ramifications of this approach to gathering provenance, and suggest some key next steps in advancing this research.

1 Introduction

Provenance and its uses have been investigated as an interesting area in Computer Science for nearly two decades [2]. In 2009, Cheney et al. describe the world of 2019 and the prominence and usage of provenance within our digital systems [3].

However in actuality, while we have seen adoption of provenance in the scientific domain, there has been a lower rate of adoption than desired elsewhere. Provenance is often difficult, expensive, or time-consuming to capture, particularly at scale in a large, distributed, technology-heterogeneous organisation. Many systems have been proposed to mitigate this, such as through aiding the direct-capture process [24, 31] or reconstructing this metadata after-the-fact [17]. However, despite their successes, this problem still persists [18].

In this paper, we examine the ramifications of relaxing the constraint that provenance must be *observed*, and present the notion of *possible provenance*. Consider a well-understood system, that is perhaps under-resourced, and as a result we have a 'snapshot' of the system in its current state, and one from several days (or weeks) ago, but with no audit logs to connect the two. While we may not be able to reconstruct the full chain of actions that led from one state to another, by lifting the constraint that we must observe changes in the system (either *in situ* or retrospectively), we can instead build a model that shows the multiple paths by which these changes may have occurred. Through this approach, we hope to provide an alternative capture method for use-cases in which possible provenance is acceptable, as well as a means of supporting resource-allocation of direct-capture systems.

The contributions of this paper are:

1. to differentiate between the concepts of observed and possible provenance, and categorise some of the existing methodologies in this space and propose how others could fit into this taxonomy in Section 2;
2. to show how possible provenance could work, in this case, with an implementation using abduction and show how these concepts work on a well-understood example — Connect Four — in Section 3;
3. to compare costs and benefits between possible-provenance approaches and observed-provenance in Section 4; and
4. to identify the open problems and considerations for future research in possible provenance in Section 5.

2 Provenance Capture

Provenance is used for a diverse range of applications, from authentication of the origin of data and entities [21, 35], to security auditing [14], to ensuring trust in decision making processes [32]. Naturally, there have been a number of approaches towards categorising different types of provenance, from the makeup of the metadata itself and whether (and if so, how) it describes the contents of the data source and/or the transformations that have been undergone [7], as well as in regard to the type of artifact, entity, or concept that the provenance is describing, and the ultimate purpose of the provenance itself [28].

In this work, we expand upon some of these categorisations, focusing on the methods of collection or (re)creation of the provenance metadata. In particular, we describe the notion of 'possible' provenance. Here, we deliberately relax the constraint that all provenance must be observed to be recorded. As such, this opens the possibility of inferred (or implied) provenance that can be constructed (or reconstructed) after the fact, not from information recorded while the system was running, but from contextual knowledge about the system itself.

Following on from this, we categorise a number of different approaches to creating and/or collected provenance metadata, both observed and possible, and describe each of the categories we have witnessed with highlighting examples. A summary of this (informal) taxonomy is shown in Figure 1. It should be noted that this is not a strict hierarchy (and certainly not prescriptive), but instead a representation of some of the observed features and approaches to metadata collection.

2.1 Observed Provenance

Our first major category is that of 'observed' provenance; that is, provenance that is recorded when a change to the system (or underlying data) is directly observed in some fashion or system events are recorded for later analysis. As such, this can be thought of as the more 'traditional' approach to provenance.

2.1.1 In-situ

The first category of observed provenance collection/creation is that which is integrated closely with the system under observation, and observes key events as they happen. This is often performed directly by the system to be observed. Alternatively, third-party tools can be applied to the system, designed to support the scientific and industrial workflow, and relieve some of the burden of having to explicitly observe and record provenance metadata as an additional format.

Some of these methods attempt to secure the necessary provenance data automatically, with as little intrusion into the workflow as possible; others rely on some form of user annotation to highlight the key parts of the system, or some explicit command, in order to cut down on the volume of data generated.

There are a number of systems that have been developed with the intention of supporting the adoption and use of provenance, and as such are capable of embedding provenance directly into their notebook system and/or workflow. For example, there have been a number of different frameworks proposed to ease the collection of provenance in scientific workflows [1, 20, 31] with the use of notebooks such as *Wrattler* [26], incorporated into the design process through *UML2PROV* [29, 30] or integrated into existing data-analysis systems such as *Imolytics* [34] and *Apache Spark* [8, 12, 33].

Another method of gathering in-situ provenance is to analyse the system directly, as it runs, and capture the key events that relate to provenance of actions and/or data. *noWorkflow* is a tool that automatically generates provenance from Python scripts with no additional input from the user or interruption to the workflow [22, 27]. Similar tools exist for other data-oriented language, such as *RDataTracker* [15, 16].

Alternatively this process can be shifted to a lower level, instead observing the containing system (e.g. the operating system), such as the method described by Pasquier et al.. This supports kernel-level observation of system events for the purposes of generating provenance information that can later be used for security-audits and intrusion detection [24, 25].

2.1.2 Post-hoc

Post-hoc provenance generation refers to situations in which events are recorded in some fashion as the system runs, but the associated provenance is not created until after-the-fact; as such, this is often the domain of third-party supporting tools.

The most straightforward approach is often to generate the provenance information by recreating the data from a log of a user's recorded actions within the system or workflow. For example, De Nies et al. use commit-logs from version-control system *git* to create the associated provenance metadata [5]. Packer et al. go on to extend this work and apply it to the popular *git* platform *GitHub*, and incorporate the platform's project-management tools (such as issue-tracking information) into the generate provenance record [23].

Other direct methods of provenance capture eschew instrumentation, and rely upon playback by performing program slicing of recorded executions. *PROV*$_{2R}$ operates at a system-level to record provenance, in part to directly alleviate some of the design-load on systems by making use of existing operating-system events [32].

In addition, McPhillips et al. use contextual system information (such as that embedded in directory structures and naming conventions), enabled by user-annotated computation blocks, to retroactively create the relevant metadata [19].

2.2 Possible Provenance

We now turn to our second major category, and what we term 'possible' provenance: provenance metadata produced without the constraint that the actions impacting the system must be directly observed.

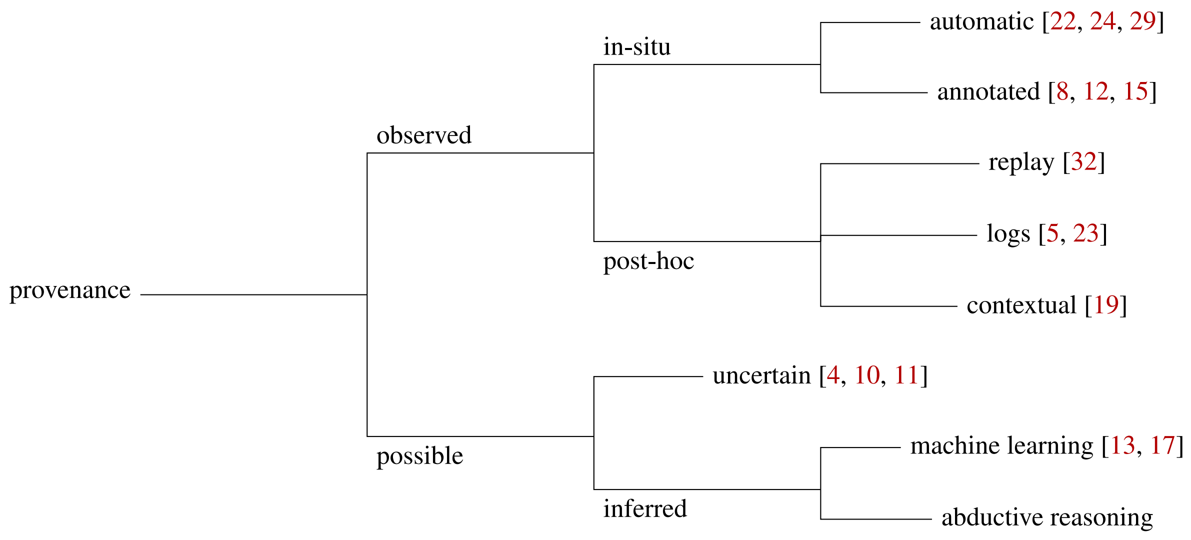

Figure 1: Dendrite diagram showing different means of collecting, generating, or inferring provenance metadata

2.2.1 Uncertain

This form of provenance can be provided with an explicitly attached measure of uncertainty; even though a particular event was unobserved, it may be possible to assume that it occurred with some probability.

Huang and Fox focus on knowledge provenance and explore uncertainty in terms of things that cannot be objectively measured but nevertheless have a bearing on the validity of provenance; they model both uncertainty in relation to trust of relationships and uncertainty of truth of facts [10]. De Nies et al. go on to extend this and provide a framework for representing this information in *W3C Prov* format [4].

2.2.2 Inferred

Inferred provenance describes the reconstruction of the provenance metadata not from logs of the system itself, but inferred based on the (likely) behaviour of the users or the rules underlying the system.

Magliacane has worked on a methodology for reconstructing provenance after-the-fact using machine-learning [17]. In this work they make use of deep reasoning to assess the most likely provenance of files in a shared-folder environment and demonstrate this working favourable in a small pilot. Kodagoda et al. use a similar approach in an attempt to infer the *reasoning provenance* (the cognitive actions) of users attempting to solve a task, using machine learning over low-level interaction logs [13]. Likewise, they report similar promising results on an initial test.

Another means of theoretically inferring provenance information from incomplete data is the use of abductive reasoning to logically expand upon the set of possible states that could be reached when following an abstract representation of the rules governing a particular system. However, in contrast to the use of machine learning, rather than necessarily making a judgement about which possible provenance set is the most likely correct one, all can be presented as candidates or collapsed into an abstraction. To the best of our knowledge, this approach has not yet been put into practice; therefore, in §3 we demonstrate an example of this type of reasoning in greater detail.

3 Possible Provenance Example

As an illustrative example of how one might infer the possible provenance of a system, we provide a worked example with the use of abductive reasoning, a form of logical inference to find the most likely explanations for a set of observations, and graph theory. With this example, we limit ourselves to a well-understood domain, to better showcase the concepts we wish to describe. For this, we use the popular board game "Connect Four".

In Connect Four, players take turns to drop coloured disks into a vertical grid; disks fall straight down, occupying the lowest empty space within the column. The winner is the first player to create an unbroken horizontal, vertical, or diagonal line of four disks of their colour.

To keep things simple enough to follow (and our diagrams legible), we use a 3×3 Connect Four grid for this example. While this raises the seemingly obvious problem that neither player will ever be able to win, for the purposes of demonstrating the provenance of moves this does not matter; it is simply the chain of moves that we care about, and this principle holds regardless of the grid size (or indeed, the system that is being modelled).

We first begin by simulating a completed game of Connect

Four to serve as the 'true' provenance, shown in Figure 2. It is not optimal play (in fact, each move is selected at random), but again, for the purposes of creating a chain of actions for us to reason over, this is ideal. This 'true' provenance is unobserved; that is, we assume that we only have knowledge of the start and end states.

3.1 Language

The first step to create possible provenance is to design the language of possible transformations. While it is obvious that it is impossible to create a language for all possible transformations, especially those driven by human-actions or one-off user defined functions, this still leaves a wide range of applications and actions that have an easy and obvious language.

To codify our system with a set of rules that can be used to infer the full graph of possible state transitions we use clingo [6] to incorporate the rules of the game itself into a logic-based inference system. We translate the rules that define the allowable transitions between states in our scenario, as well as also applying additional constraints to our ruleset that limit the generated states to fit between the known start and end states. This minimises the size of the created graph of possible provenances, which in turn allows for the simulation of more complex problems.

3.2 Expansion

In order to generate the set of possible states and transformations, we utilise the Possible Worlds Explorer (PWE) [9]. We use the PWE-framework to solve our problem by using a combination of Answer Set Programming (ASP) and Python to generate possible provenance results. Abductive reasoning is a natural fit for ASP and Possible Worlds Explorer. The result of this is shown in Figure 3.

This gives us a number of different 'possible provenance' paths; different ways that a legal chain of moves can be followed in order to reach the current state of the board. In the same fashion, this approach could be applied to — for example — a set of SQL transactions required to reach one database state from another.

Rather than suggesting what the correct provenance path may be, this approach returns all of the possible paths. While there may be a case for highlighting the most likely of the returned paths (or discounting those deemed unlikely), it is equally possible that for some use-cases it is sufficient to show whether a path was possible or not.

It should be noted that by its nature this problem-scenario naturally produces an acyclical graph; each possible node is visited once and only once (as pieces can never be removed from the board). This simplifies this problem space somewhat; however, in other domains (such as ETL) other transformations would be possible that revisit previous states and thus create cycles.

3.3 Evaluation

Once we have computed (and rendered) the set of all possible provenance paths between the known states, the question becomes "what of it?". How do we evaluate this solution as a whole, and how do we determine which of the possible paths is "closest" to the true provenance (and whether that is something that can be meaningfully defined)? To answer this, we propose the following metrics for our possible provenance graphs.

3.3.1 Specificity

We define specificity as a descriptive measure of how much of the 'true' provenance can be reconstructed and how much relies on an abstraction. For example: does our language (and the subsequent expansion) provide an (exhaustive) list of the intermediate database states? Can it determine what type of transformation was applied between each step? And/or can it go so far as to reconstruct the commands used (in SQL or an equivalent domain-specific language) to carry out those transformations? Alternatively, does the method describe a 'fuzzy' set of states and group together particular sets of transitions that are — to the user, perhaps — functionally equivalent.

For our example system, there are two types of transition between nodes (adding a red disk, or a yellow disk) and these are baked into the ruleset. As a result we can fully reconstruct the (possible) process of moves taken to reach any given state; therefore, our specificity is 100%.

3.3.2 Correctness

Correctness is defined simply as whether the graph of possible provenance contains the true provenance as a path within it. While this should always be the case (and can only be measured for certain when the true provenance is already known, for example in a testing environment), it is a useful test of the model, as a lack of correctness highlights an inconsistency, inaccuracy, or incompleteness in the language used to define the problem space.

For the Connect Four example, we can confidently say that the graph of our example model is correct as it contains the 'true' provenance (which is known to us) within it.

3.3.3 Structure

As we are representing our possible-provenance space as a directed graph, we can also show some basic information about the structure of the graph — such as the average degree, density, etc. — as a means of describing the overall network.

For example, in our Connect Four graph, there are a total of 34 nodes and 46 edges, leading to 27 different paths from the known start to the known end. The average node degree is 1.353 and the graph has a density of 0.041. The average path length is 9 transitions, which matches our expectations

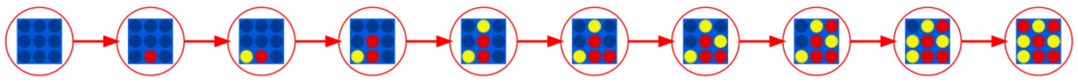

Figure 2: The ‘true’ provenance of the Connect Four game. Note that arrows represent forward movement in time, not PROV relationships.

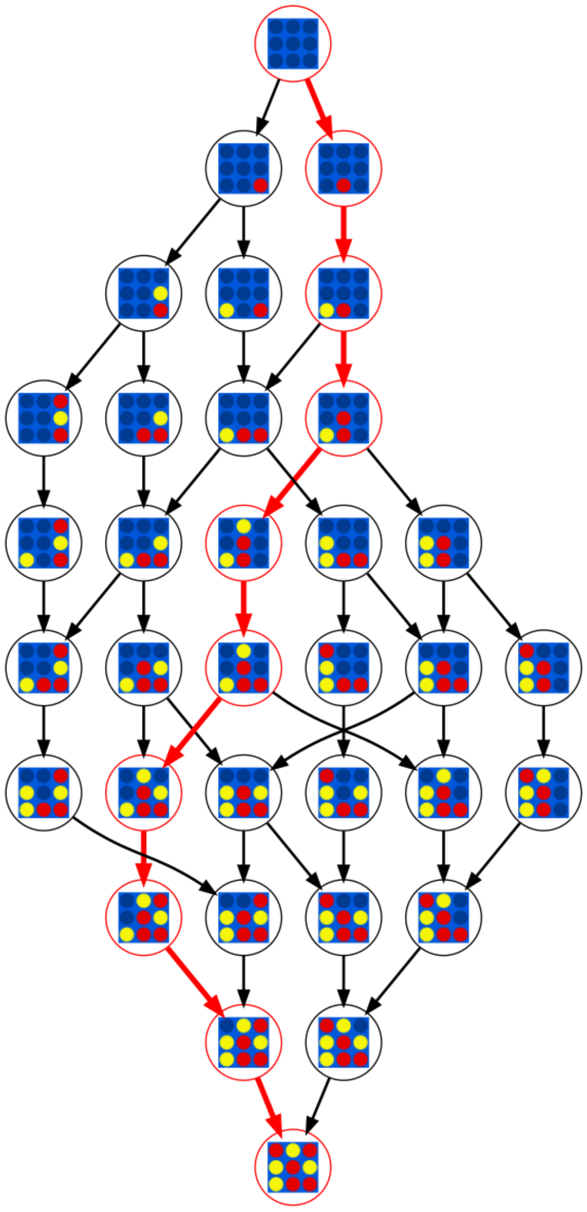

Figure 3: The set of possible provenance paths, with the true provenance highlighted

(there are 9 full cells in the end state, so every path must take 9 moves to reach it).

We can also automatically analyse the overall structure of the graph and the paths through it. Obviously, any nodes that all paths traverse *must* be part of the ‘true’ provenance (assuming our language accurately reflects the system). However, any node that has a high proportion of paths passing through it, even if not part of the ‘true’ provenance with absolutely certainty are still of value; these suggest processes within our system that, if their presence (or equally, their *lack* of presence) is observed, would assist in providing a measure of confidence to the most likely possible provenance path.

3.3.4 Accuracy

For each possible provenance path that we have generated (a path that leads from the known start state to the known end state), what level of overlap do they have with the true provenance? We define accuracy as the number of matching nodes, divided by the maximum path-length of the two paths. Therefore, paths that avoid nodes that are in the ‘true’ provenance, as well as those that contain substantial additional nodes that are not in the true provenance, will have a lower accuracy. Figure 4 shows the distribution of accuracy across each of the (non-true) possible paths for our worked example.

Because of the comparison with the ‘true’ provenance, this metric is by necessity something that we can only calculate in a testing environment. However, doing so allows us to check the accuracy of our model and analyse the consequences of any abstractions, as well as allowing us to refine our workflow based on the features common to the most accurate paths. For example, if we pick one of the possible paths, what do we lose out on compared to the ‘true’ path? Is it more complex, are there additional steps, do we lose information, and so on.

3.3.5 Variance

Variance can be thought of as complimentary to accuracy; however, rather than looking at how many nodes in each path correspond to the true provenance, we can look at how many paths pass through each node of the true provenance.

The variance for our Connect Four example is shown in Figure 5. As might be expected, the variance peaks during the middle of the path as the branches reach their most dense and complex, before collapsing again towards the end as they move closer to the known state.

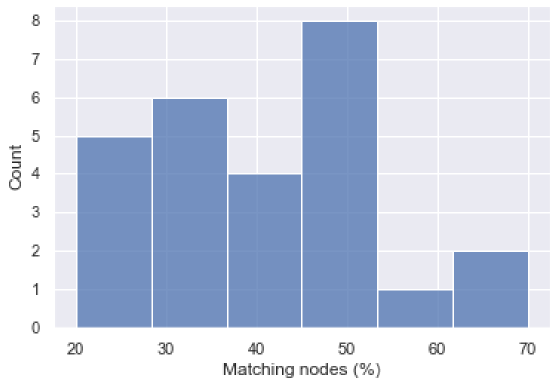

Figure 4: Distribution of Path Accuracy

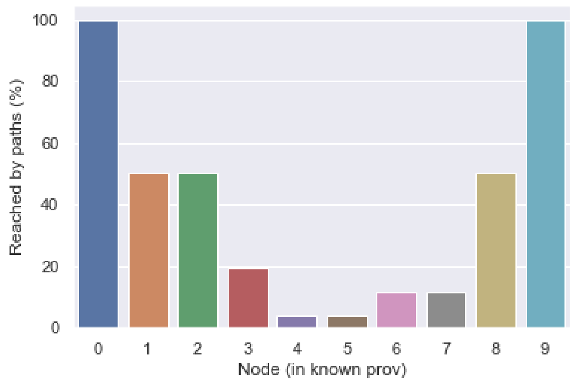

Figure 5: Path variance

As with accuracy, and for the same reason, variance can only be calculated when the true provenance is known. Again however, this can highlight features of the underlying system such as those transitions that can lead to more complex states, and allow us to ensure our model is suitable.

3.3.6 Performance

Performance is another key metric that can be measured in terms of both run-time and memory allocation; if it is infeasible to generate or reason over the space of possible provenances in a reasonable timeframe on reasonable hardware, it will be nothing more than interesting theory for the majority of users, with no practical application.

Performance is largely a consequence of the possible state space, and therefore of the overall complexity of the system to be modelled and the number of constraints that can be applied to it by the rule-set and the description language. Broadly, the more constraints that can be applied, the better the performance will be. This leads to the possibility of a trade-off between accuracy (and possibly even correctness) in exchange for performance.

3.3.7 Human Utility

However, perhaps the most important consideration when evaluating the viability of possible provenance is its human utility (to both armature and professional users). This includes an analysis of what aspects of provenance human users consider the most vital and whether our implementation sufficiently captures these features, and the acceptable accuracy when considering different use cases.

3.4 Partial Provenance

Now let us assume that we have access to a part of the true provenance; through some means we have observed (or calculated) additional known states, or ‘intermediate capture points’, between the existing known-start and known-end states.

To illustrate this using our running example, we randomly select a node from the ‘middle’ of our known provenance path to act as this intermediate capture point. This allows us to prune the graph of all paths that do not lead from the start to the capture point, or the capture point to the end, effectively breaking the problem into two smaller parts. As shown in Figure 6, this dramatically reduces the overall number of possible states.

Clearly then, the identification of suitable capture points is desirable, as this allows us to evaluate each ‘half’ of the graph individually, as two separate chains of provenance with their own known-start and known-end states. Not only does this mean that we can calculate our evaluation metrics across both halves of the problem, but doing so should ideally provide more accurate and actionable results.

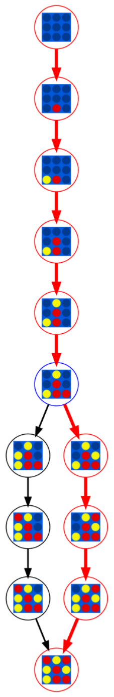

Figure 6: The provenance when we consider the highlighted Known Capture Point

This then raises the question — what makes a capture point most suitable? Not all intermediate capture points will reduce the statespace in such a way; therefore, developing a method to determine which nodes will most effectively simplify the resultant graph is key. This in turn can help us to prioritise which transitions (or combinations of,) we should make observable in our real-world system (assuming that our observations are limited by resources) in order to maximise the utility of both possible provenance and observed provenance approaches.

4 Discussion

Here we discuss some of the advantages and drawbacks of each of the approaches categorised in §2. Firstly, we can compare those methods that directly observe events to create provenance with those that instead generate possible provenance. Obviously, those that observe the provenance as it happens will produce a more accurate representation of events than those that do not. The trade-off for this is a requirement that the system either is designed in such a way as to be able to integrate provenance generation into its on-going activity, records a log of all relevant events, or mandates collaboration with the end user to annotate operational regions of note. In each of these cases, this leads to operational overheads, and often the application of contextual knowledge of the system and the domain, to determine which events are necessary to observe and record.

Within those systems that directly observe events to create provenance, we can differentiate between systems that generate their own provenance, and those that act as third-party tools or plugins for another system. While integrated tools are — by design — more convenient to the user, as their collection of provenance is embedded within their primary function, this focus also often limits them to specific domains or workflows. Likewise, we can compare between those tools that attempt to automatically create provenance data with as little intrusion and interruption of the workflow as possible, versus those that rely on user annotation and interaction in order to (potentially) have a more accurate and concise output.

Alternatively, we can compare between those systems that generate possible provenance. While none of them can guarantee that their provenance is correct with total certainty (although the use of abductive reasoning, when properly modelled, should guarantee that the correct provenance is contained within it), they have two main advantages. Firstly (as with some of the third-party approaches) they are capable of reconstructing provenance post-hoc. Secondly, by incorporating additional information into the metadata (such as uncertainty measures or possible paths) they can provide a novel perspective on the use of provenance.

Representing inferred provenance with uncertainty values embedded in the metadata is one approach. However, it may be prudent to also include, in these methods, an accompanying model of how the uncertainty was calculated or accounted for: through Bayesian probabilities, from prior user models, etc. Machine learning is another approach; with sufficiently complex set of training data, ML systems can provide a good approximation of provenance metadata. However, a notable disadvantage of this approach is generating the model; not only in the cost of time and resources to allow the machine to learn the relationships within the training set, but also in gathering the training set itself; if we have the resources to observe our system directly, we may as well use these observations to generate the provenance. However, this approach could be used to benefit existing similar systems, or to relieve resources once enough training data has been collected to 'kick-start' the ML process.

We also make the case for the use of abductive reasoning for both inferring provenance, and for informing which areas of the system are most valuable to observe to capture direct provenance. The advantages of this method are that it is wholly independent from the system itself; other than the initial and final state of the data, no output or other logging is required. The main drawback also stems from this independence: the need to logically model the system. Firstly, the task of applying the rules of the system to a logical model, and consequently, the need to abstract the rules governing more complex systems in order to maintain reasonable performance.

A key feature to consider is the eventual use-case of the provenance being collected (or generated) in any given situation. It is likely that some of these methods of creating observed or possible provenance are likely to be more appropriate for some applications over others. For example, abductive reasoning may not be useful if the end-goal is intrusion detection, as it requires knowledge of the rules governing the system (whereas intrusion often requires breaking or subverting those rules). However, it may be a better fit for domains that govern trust or ownership, or contexts in which for example the exact ordering of transactions is a lesser concern than inferring which transactions happened at all.

5 Future Work and Conclusions

In this paper we present a categorisation of a number of methods of capturing or creating provenance metadata, whether through direct observation of a system or through inferring 'possible' provenance. We demonstrate this theoretical alternative to capturing provenance of a system in-the-moment; however, rather than providing a method of reconstructing provenance we instead presented the set of possible provenances and suggest that we can facilitate the observation of "true" provenance by modelling appropriate points and conditions of the system to dedicate resources to observing. In this way, our approach can be seen as complementary to existing direct-capture methods.

A clear next step to take this work forward is to expand on the categorisation that we have performed on the various provenance-enabling frameworks; our initial work into this field is by no means exhaustive, but should provide a solid base for additional work to be built upon.

Secondly, to take the abductive-inference method described here and apply it to a 'real-world' system that would be of greater interest to the provenance community (such as an ETL Data Warehouse). In doing so it will demonstrate the feasibility of the methodology, and allow for the comparison of the proposed evaluation metrics against different variations of the system space. An important component to this work would also be applying it to a human context; how useful *is* this approach towards supporting provenance and is the resultant metadata applicable to the required use-cases? Do users — data managers, auditors, and administrators — have a use for this type of semi-abstracted provenance or at least guidance towards the most important parts of a system to observe to generate key capture-points?

Lastly, another avenue of work is to consider expanding on the theoretical approach; how much of the possible provenance is necessary, and how much of it can be 'collapsed' into abstract states? For example, within a given system, is there a particular combination or chain of events (or edges in our graph of possible provenance) that it is desirable to capture without needing to model the precise ordering? If so we can reduce the number of possible provenance paths by collapsing similar paths through the use of transitive closure.

By relaxing the constraint that provenance must be directly observed to be recorded, we allow for the possibility of lower-cost provenance gathering in an effort to support provenance capture by a wider range of tools and organisations, that is sufficiently (if not totally) accurate for their needs. In conclusion, we hope that this work helps to realise the promise of provenance by drawing together new concepts for provenance capture, and supporting more widespread provenance uptake.

Acknowledgements

This work was funded by the Infer<Proven>ence Project (EPSRC Reference: EP/S028366/1), and supported by the Federal Ministry for Economic Affairs and Energy on the basis of a decision of the German Bundestag (project number 50OO1905).

Availability

The full codebase used to generate and then infer the provenance of the Connect Four games used as an example in this paper is available as a Jupyter Notebook at: `http://github.com/InferProvenance/ConnectFour`

References

[1] Peer C Brauer, Andreas Czerniak, and Wilhelm Hasselbring. Start smart and finish wise: The kiel marine science provenance-aware data management approach. In *6th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2014)*, 2014.

[2] Peter Buneman and Wang-Chiew Tan. Data provenance: What next? *ACM SIGMOD Record*, 47(3):5–16, 2019.

[3] James Cheney, Stephen Chong, Nate Foster, Margo Seltzer, and Stijn Vansummeren. Provenance: a future history. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 957–964, 2009.

[4] Tom De Nies, Sam Coppens, Erik Mannens, and Rik Van de Walle. Modeling uncertain provenance and provenance of uncertainty in w3c prov. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 167–168, 2013.

[5] Tom De Nies, Sara Magliacane, Ruben Verborgh, Sam Coppens, Paul T Groth, Erik Mannens, and Rik Van de Walle. Git2prov: Exposing version control system content as w3c prov. In *International Semantic Web Conference (Posters & Demos)*, pages 125–128, 2013.

[6] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. A user's guide to gringo, clasp, clingo, and iclingo. 2008.

[7] Boris Glavic, Klaus R Dittrich, A Kemper, H Schöning, T Rose, M Jarke, T Seidl, C Quix, and C Brochhaus. Data provenance: A cctegorization of existing approaches. *BTW'07: Datenbanksysteme in Buisness, Technologie und Web*, (103):227–241, 2007.

[8] T. Guedes, V. Silva, M. Mattoso, M. V. N. Bedo, and D. de Oliveira. A practical roadmap for provenance capture and data analysis in spark-based scientific workflows. In *2018 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pages 31–41, Nov 2018. doi: 10.1109/WORKS.2018.00009.

[9] Sahil Gupta, Yi-Yun Cheng, and Bertram Ludäscher. Possible worlds explorer: Datalog and answer set programming for the rest of us. In *CEUR Workshop Proceedings*, volume 2368, pages 44–55. CEUR-WS, 2019.

[10] Jingwei Huang and Mark S Fox. Uncertainty in knowledge provenance. In *European Semantic Web Symposium*, pages 372–387. Springer, 2004.

[11] Nwokedi Idika, Mayank Varia, and Harry Phan. The probabilistic provenance graph. In *2013 IEEE Security and Privacy Workshops*, pages 34–41. IEEE, 2013.

[12] Matteo Interlandi, Kshitij Shah, Sai Tetali, Muhammad Gulzar, Seunghyun Yoo, Miryung Kim, Todd Millstein, and Tyson Condie. Titian: Data provenance support in spark. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, 9:216–227, 01 2016.

[13] Neesha Kodagoda, Sheila Pontis, Donal Simmie, Simon Attfield, BL William Wong, Ann Blandford, and Chris Hankin. Using machine learning to infer reasoning provenance from user interaction log data: based on the data/frame theory of sensemaking. *Journal of Cognitive Engineering and Decision Making*, 11(1):23–41, 2017.

[14] Mark Lemay, Wajih Ul Hassan, Thomas Moyer, Nabil Schear, and Warren Smith. Automated provenance analytics: A regular grammar based approach with applications in security. In *Proceedings of the 9th USENIX Conference on Theory and Practice of Provenance*, TaPP'17, page 12, USA, 2017. USENIX Association.

[15] Barbara Lerner and Emery Boose. Rdatatracker: collecting provenance in an interactive scripting environment. In *6th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2014)*, 2014.

[16] Barbara Lerner, Emery Boose, and Luis Perez. Using introspection to collect provenance in r. In *Informatics*, volume 5, page 12. Multidisciplinary Digital Publishing Institute, 2018.

[17] Sara Magliacane. Reconstructing provenance. In *International Semantic Web Conference*, pages 399–406. Springer, 2012.

[18] Sara Magliacane and Paul T Groth. Repurposing benchmark corpora for reconstructing provenance. In *SePublica*, pages 39–50, 2013.

[19] Timothy McPhillips, Shawn Bowers, Khalid Belhajjame, and Bertram Ludäscher. Retrospective provenance without a runtime provenance recorder. In *7th USENIX Workshop on the Theory and Practice of Provenance (TaPP 15)*, Edinburgh, Scotland, July 2015. USENIX Association. URL `https://www.usenix.org/conference/tapp15/workshop-program/presentation/mcphillips`.

[20] Simon Miles, Paul Groth, Steve Munroe, and Luc Moreau. Prime: A methodology for developing provenance-aware applications. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 20 (3):1–42, 2011.

[21] Luc Moreau, Paul Groth, Simon Miles, Javier Vazquez-Salceda, John Ibbotson, Sheng Jiang, Steve Munroe, Omer Rana, Andreas Schreiber, Victor Tan, et al. The provenance of electronic data. *Communications of the ACM*, 51(4):52–58, 2008.

[22] Leonardo Murta, Vanessa Braganholo, Fernando Chirigati, David Koop, and Juliana Freire. noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*, pages 71–83. Springer, 2014.

[23] Heather S Packer, Adriane Chapman, and Leslie Carr. Github2prov: provenance for supporting software project management. In *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*, 2019.

[24] Thomas Pasquier, Xueyuan Han, Mark Goldstein, Thomas Moyer, David Eyers, Margo Seltzer, and Jean Bacon. Practical whole-system provenance capture. In *Proceedings of the 2017 Symposium on Cloud Computing*, pages 405–418, 2017.

[25] Thomas Pasquier, Xueyuan Han, Thomas Moyer, Adam Bates, Olivier Hermant, David Eyers, Jean Bacon, and Margo Seltzer. Runtime analysis of whole-system provenance. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1601–1616, 2018.

[26] Tomas Petricek, James Geddes, and Charles Sutton. Wrattler: Reproducible, live and polyglot notebooks. In *10th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2018)*, 2018.

[27] Joao Felipe Pimentel, Leonardo Murta, Vanessa Braganholo, and Juliana Freire. noworkflow: a tool for collecting, analyzing, and managing provenance from python scripts. *Proceedings of the VLDB Endowment*, 10(12), 2017.

[28] Eric D Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. Characterizing provenance in visualization and data analysis: an organizational framework of provenance types and purposes. *IEEE transactions on visualization and computer graphics*, 22(1):31–40, 2015.

[29] Carlos Sáenz-Adán, Luc Moreau, Beatriz Pérez, Simon Miles, and Francisco J García-Izquierdo. Automating provenance capture in software engineering with uml2prov. In *International Provenance and Annotation Workshop*, pages 58–70. Springer, 2018.

[30] Carlos Sáenz-Adán, Beatriz Pérez, Trung Dong Huynh, and Luc Moreau. Uml2prov: automating provenance capture in software engineering. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 667–681. Springer, 2018.

[31] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A framework for collecting provenance in data-centric scientific workflows. In *2006 IEEE International Conference on Web Services (ICWS'06)*, pages 427–436. IEEE, 2006.

[32] Manolis Stamatogiannakis, Elias Athanasopoulos, Herbert Bos, and Paul Groth. Prov 2r: practical provenance analysis of unstructured processes. *ACM Transactions on Internet Technology (TOIT)*, 17(4):1–24, 2017.

[33] M. Tang, S. Shao, W. Yang, Y. Liang, Y. Yu, B. Saha, and D. Hyun. Sac: A system for big data lineage tracking. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1964–1967, April 2019. doi: 10.1109/ICDE.2019.00215.

[34] Shen Xu, Elliot Fairweather, Toby Rogers, and Vasa Curcin. Implementing data provenance in health data analytics software. In *International Provenance and Annotation Workshop*, pages 173–176. Springer, 2018.

[35] Qiannan Zhang, Tian Huang, Yongxin Zhu, and Meikang Qiu. A case study of sensor data collection and analysis in smart city: provenance in smart food supply chain. *International Journal of Distributed Sensor Networks*, 9(11):382132, 2013.