

This thesis is presented for the degree of Master of Philosophy of The University of Western Australia

Using blockchain technology to enable reproducible science



Nicholas James Pritchard

B.Sc. (with First Class Honours in Computer Science)

May 2021

School of Physics, Mathematics and Computing

Supervisors: Prof. Andreas Wicenec (Principal)

Prof. Amitava Datta (Coordinating)

Thesis Declaration

I, Nicholas James Pritchard, certify that:

This thesis has been substantially accomplished during enrolment in this degree.

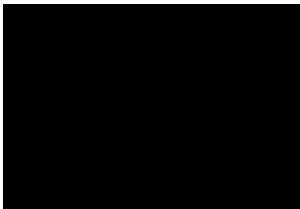
This thesis does not contain material which has been submitted for the award of any other degree or diploma in my name, in any university or other tertiary institution.

In the future, no part of this thesis will be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of The University of Western Australia and where applicable, any partner institution responsible for the joint-award of this degree.

This thesis does not contain any material previously published or written by another person, except where due reference has been made in the text.

This thesis does not violate or infringe any copyright, trademark, patent, or other rights whatsoever of any person.

Signature

A solid black rectangular box used to redact the signature of the author.

Date: July 4, 2021

Abstract

Reuse is arguably the core goal of science; humanity explores the universe by building trust in others' observations, theories, models, and experiences.

The scientific process' success has yielded technological developments that make performing high-quality science increasingly complex. The slowly resolving *reproducibility crisis* presents both a sobering warning and an opportunity to iterate on what science and data-processing entail. The Square Kilometre Array (SKA) is among the most extensive scientific projects underway and presents grand scientific collaboration and data-processing challenges. Computational workflows are an increasingly popular tool scientists use to capture data-processing tasks into concrete pieces of enacted information. The SKA, being almost entirely driven by computational workflows, poses unique challenges to scientists who must reconcile an essentially observational science with extreme-scale computing. As such, the SKA justifies a novel approach towards achieving scientific reproducibility through computational reproducibility.

This thesis frames computational workflows as a tool facilitating fundamentally reproducible science by complementing current efforts ensuring future workflow reproducibility with retroactive testing of workflow executions. This work presents a comprehensive formal definition of workflow reproducibility that is hardware, scale, software and workflow management system agnostic. Our definition extends five well-known reproducibility tenets: rerun, repeat, recompute, reproduce, and replicate into seven. A novel blockchain-inspired implementation of these reproducibility tests in DALiuGE, the bespoke workflow management system for the SKA, provides an amortised constant-time method to verify the scientific quality of computational workflows. Moreover, we demonstrate the capability to assert scientific reproducibility between a native computation and workflow management system.

Motivated by the reproducibility crisis and imminent construction of the SKA, this thesis ties computational workflow management to the Philosophy of science, integrating blockchain primitives in the process. The quest to enable reproducible science should itself be a scientific endeavour, and this work moves towards guaranteeing as much.

Acknowledgements

First, I thank my supervisors, Andreas Wicenec and Amitava Datta, whose supervision and adaption to unprecedented circumstances is genuinely exceptional and truly valued. Secondly, I thank the friends who have put up with my antics for so long. Thirdly, I thank my parents and Jess, who somehow managed to keep me from falling apart, even when the rest of the world was.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship.

Finally, I thank the reader, whoever that happens to be; I hope you enjoy the show.

Contents

Contents	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
1.1 Scientific Reproducibility	1
1.1.1 What is Scientific Reproducibility	1
1.1.2 Scientific Reproducibility from Computing Sciences	3
1.1.3 The State of the Art	3
1.2 Philosophy of Science	5
1.2.1 Logical Empiricism	6
1.2.2 Conjecture and Refutation	7
1.2.3 Normal Science and Revolutions	8
1.2.4 Scientific Frameworks	9
1.2.5 Sociology of Science	9
1.2.6 Naturalism	10
1.2.7 Scientific Realism	11
1.2.8 Summary	11
1.3 Science with the Square Kilometre Array (SKA)	12
1.4 Computational Workflows	13
1.4.1 Taverna	13
1.4.2 Galaxy	13
1.4.3 ASKALON	14
1.4.4 Pegasus	14
1.4.5 Kepler	14
1.4.6 Triana	15
1.4.7 KNIME	15
1.4.8 Moteur	15
1.4.9 Nextflow	15
1.4.10 REANA	16
1.4.11 DALiuGE	16

1.4.12	Workflow Provenance	16
1.4.13	The State of the Art	17
1.5	Blockchains	18
1.5.1	State Machine Replication	18
1.5.2	Distributed Consensus	18
1.5.3	Distributed Computing Applications	19
1.5.4	Blockchains	19
1.5.5	Iterated Blockchain Technologies	20
1.5.6	Blockchain Applications	22
1.6	Approach	23
2	Scale-Agnostic Scientific Data Processing	25
2.1	Workflow Model	25
2.2	Workflow Management System Mappings	27
2.2.1	Taverna	27
2.2.2	Askalon	28
2.2.3	Pegasus	29
2.2.4	Kepler	30
2.2.5	Nextflow	30
2.2.6	REANA	31
2.2.7	DALiuGE	32
2.3	Discussion	33
2.4	Conclusion	34
3	Reproducible Scientific Data Processing	35
3.1	Philosophical Motivation	36
3.2	Rerun	37
3.3	Repeat	39
3.4	Recompute	40
3.5	Reproduce	41
3.6	Replicate	42
3.7	Discussion	44
3.8	Conclusion	44
4	Methods	45
4.1	Cryptographic Primitives	45
4.1.1	Hashing	45
4.1.2	Merkle Trees	46
4.1.3	Merkle Directed Acyclic Graphs	47
4.1.4	Block Chains	47
4.1.5	Block DAGs	47
4.2	System Overview	48

4.3	BlockDAG Construction	53
4.4	Component and Drop Fields	53
4.4.1	Rerun	55
4.4.2	Repeat	56
4.4.3	Recompute	56
4.4.4	Reproduce	57
4.4.5	Replicate	57
4.5	Workflow Verification	57
4.6	Conclusion	59
5	Example Numerical Workflow	60
5.1	Workflow Description	60
5.2	Rerun	63
5.2.1	Method	64
5.2.2	Results	64
5.3	Repeat	64
5.3.1	Method	64
5.3.2	Results	65
5.4	Recompute	67
5.5	Reproduce	67
5.6	Replicate	68
5.7	Discussion	72
5.8	Conclusion	73
6	Conclusions and Outlook	74

List of Figures

2.1	A UML diagram depicting our arbitrary-scale workflow definition. A component is an atomic digital entity. A physical task is a single computation characterised to the level of accuracy we choose. A logical task is a single abstract computation characterised by only essential details like programming language or algorithm. One or more physical tasks realise a each logical task. A physical data artefact is a datastore like a file, distributed file system or database. A logical data artefact is a logical data resource characterised by data-type alone. Many physical data artefacts can provide equivalent data access, and a single logical data artefact may encompass many individual physical data artefacts at runtime. Scientific information comprises initial and terminal logical and related physical data-artefacts and effectively expresses the information used in making a scientific claim. A workflow is a collection of components and data artefacts with an imposed ordered structure. This ordered structure requires components and data artefacts must appear in alternating order; that is, another component cannot precede a component. A physical workflow is a workflow comprised of physical tasks and physical data artefacts. A logical workflow is a workflow comprised of logical tasks and logical data artefacts.	26
4.1	A simple Merkle tree comprised of four data elements. $H(x)$ is a hashing function for arbitrary data.	46
4.2	Example Logical Graph Template and associated hash-graph.	49
4.3	Example Logical Graph and associated hash-graph.	49
4.4	Example Physical Graph Template and associated hash-graph.	49
4.5	Example Physical Graph and associated hash-graph.	50
4.6	Example Runtime Graph and associated hash-graph.	50
4.7	Example combined hash-graph.	51
4.8	UML Sequence diagram specifying the complete life-cycle of a DALiuGE workflow with reproducibility concerns	52
5.1	UML description of our low-pass filter via FFT convolution workflow. Each library specific implementation swaps out the three physical tasks.	61
5.2	Representation of the entire low-pass filter process (Note the increase in time-series length in the final signal).	62

5.3	DALiuGE workflow for a CUDA based low-pass filter via FFT convolution created in the Editor for the Astronomical Graph Language Environment (EAGLE)	62
5.4	A UML description of our low-pass filter via brute-force convolution workflow.	63

List of Tables

2.1	Taverna to workflow model element mapping.	28
2.2	ASKALON to workflow model element mapping.	28
2.3	Pegasus to workflow model element mapping.	29
2.4	Kepler to workflow model element mapping.	30
2.5	REANA to workflow model element mapping.	31
2.6	REANA to workflow model element mapping.	32
2.7	DALiuGE to workflow model element mapping.	33
2.8	Summary of workflow management system design characteristics	34
3.1	Reproducibility tenets summarised as workflow component invariants.	37
4.1	Information stored to assert workflow reruns for different components.	55
4.2	Information stored to assert workflow repetitions for different components.	56
4.3	Information stored to assert workflow recomputations for different components.	56
4.4	Information stored to assert workflow reproductions for different components.	57
4.5	Information stored to assert workflow scientific-replication for different components.	57
4.6	Information stored to assert workflow computational-replication for different components.	58
4.7	Information stored to assert workflow total-replication for different components.	58
4.8	A summary of asymptotic complexity of each workflow operation and BlockDAG construction.	58
5.1	A comparison of workflow implementations as reruns. All but the NumPy Point-wise workflow are reruns. D1 and D2 indicate the processed dataset.	64
5.2	Averaged Normalised Cross Correlation (NCC) values for each filter implementation.	65
5.3	Workflow signatures for native and DALiuGE repeated trial executions. We truncate hash values for brevity.	66
5.4	Workflow signatures for Linux machine recomputations. We truncate hash values for brevity.	67
5.5	Workflow signatures for Windows machine recomputations. We truncate hash values for brevity.	68
5.6	Normalised cross-correlation between all filter implementations.	68
5.7	Workflow signatures for native and DALiuGE reproduction trial executions. We truncate hash values for brevity.	69

5.8	Workflow scientific replication attempts for all workflows and all machines.	70
5.9	Workflow computational replication attempts for all workflows and all machines. . . .	71
5.10	Workflow total replication attempts for all workflows and all machines.	71
5.11	Attempt to replicate experimental results in the original environment and on a different machine.	72

Chapter 1

Introduction

This initial chapter introduces pre-requisite information and supporting literature for this thesis. First, we discuss scientific reproducibility in general and computing specific contexts, framing this thesis' most comprehensive problem scope. Secondly, we introduce broad concepts from the philosophy of science, which frames scientific reproducibility against the scientific process itself. Any partial solution to such a wide-facing problem will do well to steer clear of past failings and integrate neatly with contemporary thinking. Third, we briefly reflect on the data-processing requirements of the Square Kilometre Array, narrowing down our problem scope to a particular scientific context. Next, we introduce and review computational workflows, a critical tool facilitating modern data-intensive sciences. Reviewing the development of workflow management systems establishes the state of the art in reproducible data-processing. Finally, we follow the invention of blockchain technologies from a distributed computing perspective, providing the computing context used to enact our approach to reproducible scientific data-processing. Framing blockchains as a particular type of distributed computation makes their translation to distributed computational data processing clearer.

1.1 Scientific Reproducibility

This initial section introduces the importance of scientific reproducibility in the broad scientific context and emphasises computational sciences.

1.1.1 What is Scientific Reproducibility

Reuse is arguably the ultimate goal of science; by building trust in others' observations, theories, models and experiences, humanity explores the universe. Reuse necessarily demands reproducing elements of scientific inquiry at will, be it experimental methods, published results or, more recently, computations and data [1], [2]. Growth in the scale and complexity of contemporary scientific challenges demand growth in the systems that constitute science itself. Specifics about what, how and why we aim to reproduce scientific findings is critical and different for each scientific field. Reproducibility then is a necessary part of scientific self-correction and presents a continual challenge to the scientific community to maximise scientific progress subject to novel difficulties [3]. The reproducibility 'crises' found in the fields of psychology [4], economics [5], [6], computer science [7], [8], social sciences [9], medicine [10],

[11] and education [12] to name a few is the observation that irreproducible analyses support scientific claims [13].

The lack of consensus on the practical and principal definition of reproducibility and related terminologies compound these fears [10]. Moreover, questioning the validity and widespread practices of entire scientific fields is inherently sensitive and highly complex [14].

The scientific community hold several varied definitions of scientific reproducibility. Moreover, ubiquitous access to computing fundamentally changes science and challenges traditional views on reproducibility [15]. If publications are means to announce and convince readers of a scientific claim, adding complicated software muddles clarity [15]. Claerbout and Karrenbach [16] initially applied reproducibility to computing; reproduced research meant full provenance transparency for a scientific publication including software derived information.

Reproducibility applied to other scientific fields generally refers to generating the same result data from original datasets, and replicability refers to generating the same result data from a subsequent enactment of a scientific method [14]. Goodman, Fanelli, and Ioannidis [14] define three types of scientific reproducibility; method reproduction analogous to the original definition discussed, result reproduction analogous to the original definition of replicability and inferential reproducibility, the ability to make identical scientific claims from different datasets. Fidler and Wilcox [1] discuss scientific reproducibility from a philosophical perspective, clarifying the lack of consensus present throughout the scientific world.

Most discussions of generalised scientific reproducibility focus on nuanced sociological or cognitive challenges faced when creating high-quality science. Nuzzo [17] presents a discussion of cognitive fallacies that plague study design. Bias and non-determinism undermine sound science, and unfortunately, humans are ill-equipped to deal with both [17]. In addition to heralding the announcement of a crisis, Baker [13] suggests the standardisation of methods and study pre-registration, among other suggestions, are vital to increasing experimental reproducibility. In practice, ambiguous and complex methodologies make replication difficult [18]. When processing results, expected results receive disproportionate attention leaving the vast bulk of data assumed correct due to the well-known confirmation bias [19]. Finally, Nuzzo [20] argues that scientists commonly misuse P-values for statistical significance and that they are not a gold standard for publication but the start of a more extensive investigation. Human tendencies meddle at every stage in the scientific process and is an area far too complex to summarise in its entirety. However, it seems preventing flawed science and data analysis from ever happening is a potential remedy [10], [21], but researchers pressured to output a large volume of science exasperates any potential shortcomings in the project design phase. Detecting poorly formed hypotheses is difficult in many cases [17].

Despite centuries of progress, the general process of submitting manuscripts for peer-review and subsequent publication has changed little since the 17th century, and modern scientific methods make unambiguous claims challenging to make. Scientists must prove their claim to the body of literature so that the scientific community can trust their findings. As the access to computing resources grows, a lack of consensus on appropriate information to defend a claim causes confusion and ultimately leads to poor-quality, unusable science.

1.1.2 Scientific Reproducibility from Computing Sciences

Ubiquitous access to computing fundamentally changes science and challenges traditional views on reproducibility [15]. Reproducing software or service behaviours is critical in the software industry. Fulfilling this quest has generated a vast amount of tooling, which has in practice been beneficial and has minted more than one fortune when successful. However, using software as part of a scientific process presents challenges separate from those found in the software industry. Core to this challenge is the need for scientific software to aid in making or breaking a scientific claim; sound data analysis can support poor-quality science, and flawed data analysis can muddle fundamentally sound science [22]. To this end, defining scientific reproducibility in the context of computing is far from a solved challenge. Drummond [23] initially discusses the difference between reproduction, conducting a different experiment that is scientifically equivalent and replication, the ability to perform a published computation precisely in the machine learning field. Peng [7] suggests this declaration of replication is sufficient for a minimum standard for full scientific reproducibility, but the quest to integrate computing tightly with science is left open. The role of data in science is key to this discussion, but data takes many forms, comes from many places and thus used in various ways [24]. However, it is clear and accepted that data is a first-class entity in modern science, and modern data-repositories evidence this.

Well designed scientific software packages are difficult to build and support at best, but bear great fruits when successful [25]. Encouragement for scientists to improve replicability generally revolve around improving and adhering to ‘best-practices’. Several suggestions include simple rules [26], modern industry-built tools [27], [28], the increasing use of code-sharing repositories [29] and open publishing practices [30], [31]. These improvements remain healthy, practical ways to begin integrating computation into scientific methodologies while improving confidence in scientific inquiry. Benureau and Rougier [32] provide a series of practical standards for Rerunnable, Repeatable, Reproducible, Replicable and Reusable scientific codes. Finally, Barba [33] reviews terminologies for reproducible scientific computations, finding that a lack of consensus is still present and that governing bodies may hold conflicting definitions.

We shall see, recent developments from the scientific community highlight both the difficulty of creating contemporary science and integrating computational methodology across a wide range of fields.

1.1.3 The State of the Art

After the realisation and formalisation of a contemporary, wide-reaching scientific reproducibility problem, we present several attempts to refine the issues at hand further and address them. Munafó, Nosek, Bishop, *et al.* [34] notably present a complete description and several suggested remedies to the reproducibility crisis. Munafó, Nosek, Bishop, *et al.* [34] cite the increasing difficulty in separating data-noise to ‘true’ scientific fact as the fundamental issue and propose initiatives reaching across all aspects of the scientific process from technical improvements to methodology, trial pre-registration, pre-print services, open-access tooling and re-incentivising the peer-review and publication process. Ioannidis [35] introduces the burgeoning field of meta-research, applying scientific methodology to improve scientific processes and clarifying that keeping scientific quality high is a consistent challenge made more difficult with science’s continued success. Peng and Hicks [22] reflect on development in

computational human biology, concluding that reproducibility is a continued effort unsuitable for a publication first approach; reusing code and data may occur long after an initial publication. Nissen, Magidson, Gross, *et al.* [36] create a Markov model for confidence in scientific claims, concluding that without a sufficient number of negative, published results, a false claim could become canonised as fact. However, it is essential to temper quests for ever-impressive science with historical context; irreproducibility has always plagued science. Shiffrin, Börner, and Stigler [19] address the paradox that scientific success seems to create more problems with science itself and suggest this is unsurprising since extending the ever-growing body of scientific knowledge ventures further into the unknown and is hence prone to error. Suggested remedies follow a growing trend towards re-incentivising the publication system and study pre-registration. In a staggering study involving 197 researchers, Botvinik-Nezer, Holzmeister, Camerer, *et al.* [37] task 70 research teams with an identical research task testing nine ex-ante hypotheses against the same fMRI dataset, providing practical evidence that analytical flexibility has a significant impact on scientific findings and suggest reducing researcher degrees of freedom via trial pre-registration, sharing analysis workflow toolings and running datasets through several analytical workflows as potential improvements towards achieving practical scientific consensus. An increasingly common remedy to improve scientific reproducibility in various fields is the concept of trial pre-registration [38]. Pre-registered reports concretely define data collection, subject recruitment and analysis mechanisms, among other details but most importantly, provide a structured method to deal with unexpected discoveries. Pre-registration is increasingly common in physical sciences [34] and encourage making scientific processes well-defined pieces of factual information, now subject to information processing.

The prevalence of computing in science combined with contemporary scientific reproducibility concerns presents several opportunities to integrate computing into science better. In a balance of rigour and practicality, one generally separates results and computing elements and address each separately. There is a growing emphasis on data reuse in an age of information explosion [39]. Reusing data, in particular, depends heavily on the initial source’s quality and encouraging such reuse is proving particularly difficult to achieve. A significant part of this issue is finding a suitable way to index, store and map out the world of open data. Bellini [40] investigate the feasibility of a blockchain-based dataset identifier (similar to a DOI) as a means to make data a first-class and citable entity.

Part of trusting and reproducing data lies in trusting and subsequently reproducing the process of generating said data; a simple separation into results and computing elements is not always sufficient. Lamprecht, Garcia, Kuzak, *et al.* [41] integrate the FAIR guiding principles for digital research [42], typically applied to data, to research software, proposing extensions where appropriate. Lamprecht, Garcia, Kuzak, *et al.* [41] notably discern the lack of a global identifier system for software where such systems for publications and data exist. Konkol, Nüst, and Goulier [43] investigate 15 infrastructures for facilitating reproducible scholarly communication. They find a trend towards requiring literate programming and a lack of proper versioning between a published and latest resource are potential problems. Computational workflows are a popular method to describe and enact complex scientific computing tasks with many systems available [44]. Considering workflow reuse support beyond initial publication, Beaulieu-Jones and Greene [45] automate the testing of a computational workflow using Docker and continuous integration, two methods from Software Engineering. Sandboxing all

computing tasks facilitates reproducibility, and testing any component changes against known results ascertains introduced errors. Feger [46] extensively investigates the relationship between researchers and reproducible computing tools regarding the scientific reproducibility efforts in the field of High-Energy Physics [47]. Considering how researchers (people) ultimately integrate any proposed solutions into the practical process of science is critical for any significant benefit, the trend towards required programming competence discovered by Konkol, Nüst, and Goulier [43] not boding well in this regard.

Finally, the very definition of scientific reproducibility concerning computing is of utmost importance. Gundersen [48], in addition to re-framing scientific inquiry as a machine learning task, provides an excellent review of reproducibility and replicability definitions. The definitions follow several trends:

- Most require identical methodology.
- Method implementation is a common difference between reproduction and replication.
- Some bodies require identical hardware to reproduce a work.
- Data is firmly a part of reproducibility, and finally, result interpretation is uniformly a constant requirement for reproducibility and replicability.

Notably, all definitions surveyed require consistent scientific theory and hypotheses. Few innovations have served to improve humanity’s scientific capacity greater than computing. Perhaps the scientific community’s reproducibility crises are not a crisis per se but the growing pains symptomatic of the scientific community learning to use more sophisticated tools of inquiry. We now introduce a brief history of fundamental scientific thinking to frame the challenges facing modern scientists.

1.2 Philosophy of Science

By reviewing developments from the philosophy of science, we hope to frame reproducibility from first principles, avoid some pitfalls of the past and ultimately provide a fundamentally sound innovation. We must therefore understand why reproducibility is core to the scientific process. This section is a rapid departure from contemporary issues in science to a brief recount of a century or so of scientific thinking. We present several historical approaches to the philosophy of science in a roughly chronological order inspired by Godfrey-Smith [49], whose text we encourage the motivated reader to pursue. For each significant approach covered, we describe its main features and potential shortcomings with several sources for those needing more information. All approaches presented are initially quite reasonable with distinct merits. However, we shall see their faults appear under scrutiny.

Additionally, debating science’s fundamental philosophy is far from a solved problem, including what a theory about science should even encompass. However, the ability to discern scientific from non-scientific inquiry and seeking a balance between abstract theory and rigid processes are central motifs in the literature. We do not discuss such debates here, preferring to spectate on the last century or so of developments.

1.2.1 Logical Empiricism

Logical empiricism is an approach to science grounded in human experiences dating from the seventeenth and eighteenth centuries. Contemporary rationalists such as Descartes and Leibniz believed the world could be understood purely through reasoning, as is the case in formal sciences such as mathematics. Empiricists such as John Locke and David Hume persisted that experience is the only source of knowledge. Immanuel Kant [50] stands as a notable exception taking a sophisticated intermediate approach based on logically organised experiences. Logical positivism of the early 20th century is an approach to scientific philosophy based on conveying experience through language. This theory centred around two ideas, the distinction between analytical and synthetic sentences and verifiability. The meaning of analytical sentences is constant regardless of the state of the world. The meaning of synthetic sentences depends on the state of the world. Verifiability refers to how words inscribe meaning, according to logical positivism in its verification or observable effects. Logical positivists believed they could translate all sentences into patterns of observable effects. While initially not upsetting, language that intends to describe real meaning fails to do so, such as theoretical statements with only indirectly observable phenomena (like atomic theories).

Logic is, unsurprisingly, a central component of logical positivism and empiricism. Deductive logic is uncontroversially a valuable tool for logical empiricism, giving the reasoning behind what makes arguments compelling. Inductive logic, however, is much more problematic. Inductive logic attempted to incorporate real-world phenomena into a logical structure but always allow for unexpected results. Balancing the logical structure needed to assure future statements and tolerating novel discovery which upset these assurances was ultimately damning to the movement.

Problems with Logical Positivism

Problems arose with logical positivism and later logical empiricism around the central components of a verifiability principle and formulating a satisfactory inductive logic. While the initial ideas presented in this era are amenable to many contemporary scientific discoveries, these ideas ultimately fall down under scrutiny. Verifying simple statements with testing is an intuitive approach to science. However, adding logical structure to these statements added significant complexity. Quine [51] argues, against logical positivism, that all testable statements, even simple ones, are holistic and complex networks of claims and assumptions about the world; one must absorb an entire testing context to accept any observation. Treating experimentation effectively as a long series of conjunctions leads to unexpected results owing to experimental failure or incorrect assumptions or legitimate discovery becomes impossible. This approach contradicts the aim of logical positivism to reduce scientific inquiry into atomically verifiable statements. Moreover, a holistic approach incorporates a web of statements, some analytic, some synthetic in the positivists' sense makes a sharp distinction between the two types of language inconsequential since they occupy the same complex web of experience.

Quine's arguments impacted the logical positivist movement which went through subsequent reform. The resulting argument asked to accept a baseline set of statements about the world at a given time and to use empirical experiences to modulate only a small part of a complex network of statements. As science solved increasingly sophisticated problems relying on increasingly difficult to observe phenomena (genes, electrons, quarks), defending this position became increasingly difficult.

Inductive logic is problematic when attempting to use experiential evidence to support a logically constructed scientific theory. This problem arises repeatedly, but the logical empiricists' approach to formalise a relationship between evidence and confirmation was ultimately doomed. The intuitive idea of hypothetico-deductivism, which comes naturally to many people and still taught at a high-school level, suggests one should trust that theories backed by increasing evidence. Such an approach is practically acceptable, but after how much evidence precisely can one confirm a theory? How can we logically deduce the 'next' observation while maintaining a possibility to be wrong? The well-known ravens problem and Goodman's riddle of induction [52] are examples of such foibles.

In the meantime, throughout the 20th century, a more workable and more sophisticated approach towards describing science arose, displacing first logical positivism and later logical empiricism as the dominant approach towards theorising science itself. However, the role of experimentation, logical and structured approaches towards evidence gathering remain central components of modern science. Discussions surrounding logical positivism guide our thinking on what information is important to make a computing-based scientific claim, defining what 'experience' means in a sea of data processing.

1.2.2 Conjecture and Refutation

Karl Popper is a well-known, highly influential contributor to the philosophy of science whose straightforward approach to describing science resonates with many working scientists. While technically part of the empiricist movement, Popper's goal was to provide a method to demarcate scientific from non-scientific theories. Popper's solution is 'Falsificationism', claiming a hypothesis is scientific if and only if some possible observation has the potential to refute it [53]. Inductive logics surrounding confirmation were of no use to Popper's theories, preferring the position that passed observational tests do not increase scientists' confidence in a theory. The job of scientists is to propose bold claims and provide clever means to try and refute them. When an experiment supports a theory, a scientist 'has not yet falsified the theory'.

Popper's approach to handling scientific change is an elegant and endless cycle of offering bold conjectures and clever refutations. This heroic depiction of science as a simultaneously creative and pragmatic endeavour appeals to scientists, but the more interesting idea is to build an open-minded scientific team out of close-minded individuals focussing on either conjecting or refuting. This concept preludes later social approaches towards the philosophy of science.

Problems with Popper

Popper's theory of conjecture and refutation, while pragmatic and straightforward, has some considerable problems. Firstly, an approach that labels theories as scientific or not is probably not nuanced enough to capture the world's complexity. The holism problem that plagues logical empiricists strikes again since we must ultimately decide if a falsification sufficiently refutes a conjecture. Moreover, rejecting a theory as un-scientific is itself a falsifiable decision. Hypotheses that make improbable but possible claims are similarly problematic; if one claims such a hypothesis as insufficiently bold, at what point is a hypothesis successful? Alternatively, one could define an arbitrary probability deemed 'too unlikely', a practical if problematic approach. Finally, how does one choose a theory to solve a practical problem? If all our scientific philosophy permits is falsification, there is no reason to trust

an old, tested theory over a new, untested one. Popper’s solution is ‘corroboration’, which is subtly different to confirmation by observation where one can effectively recommend a theory.

Ultimately, Popper’s theory of conjecture and refutation is philosophically problematic as an all-encompassing philosophy of science since it can falsify a theory without a deductive logical relation between observation and theory, making scientific claims ultimately arbitrary decisions. Nevertheless, the powerful simplicity of conjecture and refutation is an integral part of our approach; much of contemporary computing depends on ultimately arbitrary decisions. This one-step process to build scientific experience naturally extends itself to modern software testing.

1.2.3 Normal Science and Revolutions

Aside from Karl Popper, Thomas Kuhn is possibly the most widely influential figure in the philosophy of science. In ‘The Structure of Scientific Revolutions’ [54], Kuhn does not describe science as a philosophy of rational inquiry based on logic but as a social phenomenon. More specifically, Kuhn characterises scientific work as a series of two phases; a period of ‘normal’ science dominated by a single paradigm, followed by a period of crises in which a few or a single piece of extraordinary work resolve. These works in turn define the next paradigm and period of normal science. Kuhn uses the phrase ‘paradigm’ in various meanings, but the most common and notable in a broad sense refers to a way or methodology of doing science in a particular field and, in a narrow sense, refers to exemplar achievements that define a broad paradigm. Examples of exemplary works include Mendel’s peas, Newton’s laws of motion or Maxwell’s equations. A given paradigm provides the structure of normal science, and the lack of a paradigm causes unstructured and chaotic crisis science. Methodological and linguistic consensus organises scientists’ efforts allowing their collective problem-solving power to pool together. Kuhn suggests the existence of a pre-revolution period of crisis where a paradigm fails to cover results discovered by normal science. A crisis period preludes a paradigm shift, where the total problem-solving power effectively outstrips ‘normal’ methodologies. The ability for science to be at times ordered and in others broken down and reconstituted is a central feature contributing to science’s long-term success.

Kuhn motivates his characterisation by reflecting on historical, scientific developments, an approach that proved challenging for logical empiricists. Kuhn stresses that it is often individual scientists’ quirks that affect science’s success when moving between paradigms during crisis-science. This makes Kuhn’s approach decidedly more normative than the descriptive theory logical empiricism provides. Like Popper, Kuhn mixes open-minded creativity with structured pragmatism but does so over longer periods, rather than in each experiment. Any approach to enable reproducible data-processing must accept this creative trend and dynamism rather than be a static recipe for science.

Problems with Paradigms

While more in line with how science practically works, there are many issues with Kuhn’s approach. The infamous and extensively discussed ‘Chapter X’ of Kuhn’s seminal work presents Kuhn’s most radical thoughts, most notably, a change in paradigm changes the fabric of reality itself [55], [56]. The depiction of science as periods of ordered progress and chaotic revolution aligns nicely with several major scientific developments but is not wholly satisfying. There are some other issues with Kuhn’s

formulation. Firstly, fields are rarely genuinely dominated by a single paradigm at a time, an idea we revisit soon, and a crisis has rarely accompanied significant paradigm shifts. However, as the first to formulate science as a transient cultural phenomenon across societies and history, the concept of scientific paradigms resonates to this day and is an essential step in the philosophy of science.

1.2.4 Scientific Frameworks

Kuhn sparked a fierce debate around the structure of science. This sub-section introduces the systems Imre Lakatos, Larry Laudan and Paul Feyerabend, three of the most notable post-Kuhn commentators, describe.

Imre Lakatos worked with Karl Popper, saw Kuhn as an agent of chaos and sought to reform Kuhn's ideas with the concept of research programmes [57]. While roughly similar to Kuhn's paradigms, Lakatos' scientific programmes comprised of a stable inner-core of accepted theories and an outer-belt of applied and expanding science (not unlike Popper's conjecture and refutation). Research programmes are initially a reasonable model, but a key point is missing; how do scientists establish new research programmes?

Laudan [58] iterates on Lakatos' programmes but adds nuance. Laudan wished to address the competition between paradigms and presents a more cohesive depiction of science during this process. Theories hold logical precedence over each other where scientists group them into 'research traditions'. Scientists can move between traditions, thus capturing a key proponent of science which is why scientists embark on work that may not yet be entirely accepted. Thus, it is rational to pursue currently unaccepted research traditions if they have a high progress rate. In a critical oversight, Laudan does not consider how decision making changes in a collection of scientists.

Finally, Feyerabend and others [59] argued for 'epistemological anarchism' based on historical accounts. Science historically comes from opportunistic and creative individuals without regard for anything resembling structure. A central example is Galileo's persecution, whose alternate view of the cosmos violated any prior experiences and fought against the accepted beliefs of the time. To Feyerabend, science is about challenging and not following observations, and while his ideas often stray far into chaos, Feyerabend admits structure is sometimes necessary.

Process Driven Theories of Science

So far, we have seen one-process depictions of science in logical empiricism and Popper's process of conjecture and refutation. Later Kuhn, Lakatos and Laudan formulated two-process theories that dominated the 20th century. Finally, Feyerabend argues for the use and quick rejection of structure and frameworks but are not a necessary scientific characteristic. We now briefly review an entirely different approach to approaching a philosophy of science.

1.2.5 Sociology of Science

Science is a social enterprise of many individuals working together. As such, Sociological approaches to science also grew support. A central figure, Merton [60], characterised science with four norms.

- Universalism - The irrelevance of personal attributes.

- Communism - The common ownership of scientific ideas.
- Disinterestedness - A more questionable idea that scientists work to a common goal rather than personal gain.
- Organised scepticism - The community-wide pattern of challenging and testing ideas.

Moreover, Merton argues the primary incentive mechanism in science is recognition for new ideas. This incentive mechanism is compelling when considering current science issues such as a ‘publish or perish’ mentality or the reproducibility crisis. Merton’s ideas are well-founded and go a long way to explain how scientists operate. However, more sophisticated formalism have recently succeeded Merton’s.

A major successor is Barnes and Bloor’s ‘Strong Program’ [61], which treated scientists just like any other group of people. The Strong Program also sought to analyse scientific theories in the context of their social circumstances. However, this type of inquiry is complex and often controversial. Shapin and Schaffer’s ‘Leviathan and the Air Pump’ [62] develops these ideas following scientific processes as we know them (lab groups, publications, societies). Latour and Woolgar’s ‘Laboratory Life’ [63] follows a Nobel-prize-winning group’s daily life, making the principal point that facts are made as part of a process and not found.

Recognising that science a sociological process driven by individuals collaborating and endeavouring together just like any other part of humanity opens up several ways to view well-known scientific phenomena and has since been a subject of scrutiny. In many respects, the reproducibility crisis and burgeoning field of meta-science are contemporary extensions of this line of inquiry. Therefore, any partial solutions or remedies must also account for the sociology of science. However, a purely sociological view is not enough to explain how theories, observation and testing fit together to help humanity solve practical problems.

1.2.6 Naturalism

Naturalism is a formulation of science that argues science is a tool used to frame ourselves in the wider world. Directly applying philosophy dooms any attempt to characterise science since philosophy is an artefact of natural beings, and therefore part of the system it is trying to describe. Quine and others [64] established naturalistic scientific philosophy. Additionally, theory biases all observational evidence; this is a characteristic of scientific inquiry.

Hull [65] formulates naturalistic science as a social process. Starting with the intuition that science results from naturally curious individuals combining their efforts, formulating science as a collection of empirical testing is natural. The unique combination of cooperation and competition contributing to science’s continued success results from the reward system of recognition. A self-interest to direct one’s efforts efficiently and generate goodwill motivates replicating others’ work, but checking happens rarely. Oldenburg’s peer review system, the first and still current method of publishing formal scientific works, is the first broadly accepted start of this quest for individual recognition. Ultimately naturalism seeks to resolve philosophical questions about the world with a scientific approach to placing ourselves in our world. While still based on empirical testing, unlike logical empiricism, naturalism starts with no philosophical structure, instead of describing how the social structures surrounding science affect this quest for knowledge about the universe.

1.2.7 Scientific Realism

Scientific realism is one last approach we briefly discuss, seeking to address the ever-present disconnect between the natural world and its descriptions. Science, then, is our strategy for refining and improving our descriptions, theories and approaches. The confidence we place in our descriptions changes depending on the specific field we consider. Ultimately, we alter our descriptions by carefully constructed observation. An obvious question is how much evidence do we need to be confident in a particular theory? Bayesianism [66] is one such approach, seeking to frame hypotheses in a probabilistic framework. While appealing for its mechanical approach, a Bayesian framework is only a practical and valuable toolset in fields where such an expression is natural. Scientific realism suggests that we trust in science and the scientific process as a means to build a helpful description of the universe around us.

1.2.8 Summary

We have briefly reviewed several approaches to describe the scientific process and attempts to frame what makes scientific knowledge successful. Logical empiricism tries to encompass all that is scientific into the language of scientific knowledge. Doing so is ultimately impossible since inductive reasoning cannot reasonably account for future observations. Popper [53] formulates science as a one-step process of conjecture and refutation. This approach is too simplistic to be wholly satisfying but is simple enough to be widely appealing to the pragmatic. Kuhn [54] frames science as a series of broad paradigms, including social structure considerations into accepted knowledge in a given paradigm. This two-step process of normal and crisis science is unacceptable since it is not entirely consistent with history and contemporary practice. Lakatos [57] and Laudan [58] each provide refinements on the paradigm concept. Feyerabend and others [59] reminds us never to forget our ability to reject any such framework if it no longer fulfils our needs. Sociological approaches [60]–[63] fair well to explain why scientists may behave the way they do but say relatively little about why we should trust scientific knowledge. Naturalism casts science as a way to answer inherently philosophical questions about our interactions with the natural world. Scientific realism frames science as the only responsible way to characterise this natural world.

Despite these discussions, learning from experience sits at the core of the scientific approach; empiricism hangs around. Moreover, objective observation combines with a complex social structure to provide an overall strategy for exploring the world. The balance between competition and cooperation in this social structure is a tenuous but critical contribution to science's success. The exact structure of science is likely too complex and diverse to satisfyingly capture, but some key learnings become apparent:

- Science demands a balance of creativity and pragmatism.
- Simplistic recipes to conduct science often fail.
- Attempting to assert facts about future observation based on experience is a scientific challenge.

Addressing discourse in the scientific process is an inherently scientific endeavour. Determining where to address existential challenges to science's efficacy in the social structure or observational methodologies is a continuous discussion. However, science is only ever sure about past and not

impending observations, a core learning guiding our approach to enabling reproducible science in modern data-processing scientific projects.

1.3 Science with the Square Kilometre Array (SKA)

The SKA is the proposed world’s most extensive radio telescope array comprised of approximately one square kilometre of collecting area when fully implemented and the necessary infrastructure to process its observations. Conceived in the 1990s and built across Australia (Murchison Widefield Array) and South Africa (Karoo), the SKA is one of the most significant singular science projects in existence [67]. Even after its initial phase I construction, this machine would provide an order of magnitude increase in accuracy and scale, facilitating novel and world-class science. Moreover, the SKA is one of the most significant international collaborations to date, spanning institutes from 21 countries. Building the SKA and its related infrastructure demands grand-scale collaboration between academia, industry, and government partners. The SKA promises world-class science by building world-class facilities. Reproducibility and related concepts are constituent to this goal and at SKA-scale provides the opportunity to innovate in this space.

Braun, Bourke, Green, *et al.* [68] outline the expected fields of science the SKA addresses. Understanding the cosmic dawn, the epoch of galaxy formation, and the current epoch of acceleration are three original motivators for constructing a SKA-type machine. All SKA science goals require scaling from precursor facilities to SKA phase 1 (SKA1) and finally full-scale (SKA2) facilities. Asserting reproducibility in early-stage science should ease the challenge of growing into full-scale resources. Scenarios where similar phenomena are possible to witness with other equipment [69]–[71] make a strong case for reproducing known results between teams.

In other areas of interest, the SKA provides vast increases in sample numbers (pulsar science) that require automatic or ‘robotised’ and faithful method replication. Another example is the observation of transient events [72]. Not only will transient observations leverage robotised methods, but the deferral of other longer-term scheduled observations require verifiable real-time adjustments that cannot harm scientific quality. Another case for building high confidence at a tool level is the highly marketable search for extra-terrestrial life (SETI) [73]. While highly exploratory, any truly novel findings should rightfully attract the highest levels of scrutiny. Building confidence around data points at the most fundamental level is essential to quality science and so including reproducibility into SKA tools improves the quality of the SKA’s scientific output and therefore trust in its final scientific findings.

Grand-scale science projects place unique, but not entirely unseen, challenges for scientific quality. High energy physics (HEP) and associated institutes such as CERN provide opportunities to learn from fields working with relatively unique hardware. Technical innovations created to operate the Large Hadron Collider (LHC) is no small contributor to its continued success [47]. The compute requirements and scale of the SKA and LHC are similar; however, the real-time nature of continuous observation builds a strong case of building reproducibility efforts into the SKA infrastructure itself [74].

Ultimately, vastly complicated, highly distributed computational workflows produce the SKA’s scientific output. By targeting reproducibility at this level, we move towards a fully reproducible science endeavour. Facilitating high-quality, reproducible data processing workflows moves the SKA one step closer to producing high-quality, reproducible science. Not only will this provide confidence in

the academic findings of this enormous scientific project but promote its good practice and provide an example for future projects. As such, by building a solution intended to assist scientific reproducibility generally and be commensurate with modern thinking about the scientific process itself, we hope that the SKA contributes to the scientific process itself, in addition to the scientific contributions of its own.

1.4 Computational Workflows

Scientists are not software engineers, and nor should they be. Modern scientific endeavours demand increasingly sophisticated and complex data processing. Computational workflows connect abstract descriptions of computational processes and resources, forming aggregate computational methods. Workflow management systems permit scientists to connect their domain-specific knowledge and needs to computational resources flexibly and reusably. In the past two decades, workflow use has matured, and the variety of systems developed, deployed and iterated upon in this time offer unique characteristics, idiosyncrasies and use-cases. In this section, we introduce several unique workflow management systems. Investigating the history and state of the art in workflow management systems allow us to approach reproducibility in a widely applicable manner later. We note that this review primarily focuses on traditional directed acyclic graph based workflow management systems rather than a total review of all computational workflow management systems that take different operation modes.

1.4.1 Taverna

Increasing reliance on *in silico* experiments in the bioinformatics community motivated the development of Taverna [75]. Many have repeatedly reworked and adapted Taverna for other domains such as Astronomy [76]. Scientists assemble atomic computing tasks through a graphical user interface, constructing workflows in a bespoke language; Scufi. The contemporary maturing of Internet hosted computing resources influences Taverna's reliance on web services, data-stores and applications hosted online and invoked via well-formed messaging protocols [77]. Focussing on remotely accessible resources encouraged adoption by a wide range of third parties and thousands of participating services, promoting programmatic interoperability but introducing complexities when 'shimming' services together [78]. Data-volume growth encouraged a re-development of Taverna to include parallel execution, separated data and processing spaces and additional high-level workflow logic permitting more complex workflow execution [79]. Arguably, however, the chief contribution of Taverna is the abstraction of a workflow in itself, encouraging public sharing of concrete protocols and spawned a new era in data-centric science [80].

1.4.2 Galaxy

Science is a highly exploratory and often creative process; Galaxy [81] is a web-browser based genomics analysis toolkit focussed on connecting non-programmers to computational resources. A workflow in this domain is effectively a complex series of database queries and is, like Taverna, based on connected web services but can integrate with any command-line tool. Abstracting query workflows encourages

more expressive method descriptions and sharing beyond raw scripts through galaxy pages [82]. Galaxy has enjoyed wildly successful community support but still faces scalability issues [83]. Moreover, while Galaxy’s simple tool interface plays a large part in the continued support and integration of new tools, it is not easy to guarantee long-term workflow functionality [84], a constant balancing act between flexibility, performance and transparency. Nevertheless, Galaxy is among the most popular and successful scientific workflow management systems available.

1.4.3 ASKALON

Askalon is a workflow management system designed to construct grid-based applications [85]. There is less focus on connecting scientific domain knowledge to computing resources and focusing on innovative workflow design. Askalon workflows use AGWL [86], an XML based modelling language. Teuta is a program for graphical workflow composition and exposes a UML-like interface to AGWL workflows [87]. Using such a strictly defined language allows Askalon to schedule tasks across various resources, orchestrating highly scalable workflow executions [88], [89].

1.4.4 Pegasus

Like Askalon, Pegasus strives to pursue otherwise technically infeasible workflows [90]. Unlike Askalon, however, Pegasus conforms to a strict directed-acyclic-graph workflow model [91]. Pegasus terms atomic computing tasks jobs, which in addition to files and dependencies form workflows. A core goal is instantiating highly parallelised workflows on a variety of high-performance computing environments with a particular emphasis on reliability, and efficiency [92], [93]. By utilising a highly programmatic approach towards workflow scheduling, Pegasus consistently pushes the envelope of workflow performance.

1.4.5 Kepler

Ludäscher, Altintas, Berkley, *et al.* [94] introduce Kepler, a workflow management system aiming to build scientific workflow applications out of relatively tightly coupled components. Kepler’s unique system of actors, parameterised processes applied to data, and directors, processes orchestrating the dependencies between actors make up workflows. The unique actor-director structure allows easier concern separation and component replacement. Novel at inception, Kepler natively includes provenance collection, capturing work done and execution context through adding a director-like object [95]. Besides improving scientific integrity, provenance information adds a technical use similar to build systems in traditional software development; re-running workflows only re-executes components with changed parameters. Deelman [96], an original member of the Pegasus development group, echoes the need for workflow provenance information to further workflow use. Bowers, McPhillips, Riddle, *et al.* [97] expand upon Kepler for phylogenetic data analysis specifically, with particular care for provenance tracking. Unlike Taverna or Kepler’s original provenance collection to improve workflow performance, Bowers, McPhillips, Riddle, *et al.* [97] focus on data-oriented scientific provenance.

1.4.6 Triana

Triana is a data analysis framework aimed at developing workflows visually [98]. Triana workflows support heterogeneous grids and a peer-to-peer communication model, similar to Kepler [91]. Also similar to Kepler, Triana instances processing elements from Java components.

1.4.7 KNIME

The Konstanz Information Miner is an open-source data pipeline assembler aimed at both science and business use [99]. KNIME has well-formed node forms and is therefore quite extensible but perhaps not built for extreme-data scales required by some environments.

1.4.8 Moteur

Based on the observation that many scientific workflow management systems exist, interoperability between them is lacking; Glatard, Montagnat, Lingrand, *et al.* [100] present Moteur. Moteur simplifies workflow expression while maintaining or improving execution performance. The distinction between task-based workflows and service-based workflows is particularly troublesome. The GWENDIA language powering Moteur leverages array-programming concepts to be expressive, data-driven and asynchronous. Other workflow languages offer a tradeoff between expressiveness (Scufl in Taverna) and performance (DAGMan in Pegasus) but specifying a language separately from its enactment allows workflow execution on multiple platforms [101]. Balderrama, Huu, and Montagnat [102] utilise Moteur to combat the low reliability of large-scale distributed computing facilities, the performance overhead of shared infrastructures, the balance between user jobs and application complexity by wrapping workflow components as services and combining local and distributed execution models. Rogers, Harvey, Huu, *et al.* [103] exploit the developing tendency for workflow management systems to utilise common workflow representations to achieve inter-operable workflows. Finally, Plankensteiner, Prodan, Janetschek, *et al.* [104] bring interoperability between European workflow management systems (ASKALON, Moteur and Triana). Translating abstract workflow descriptions and bundling this information with concrete component descriptions achieves workflow interoperability. Maturing workflow management systems begin to make good on their promise to generalise scientific computing efforts.

1.4.9 Nextflow

Di Tommaso, Chatzou, Floden, *et al.* [105] present Nextflow, a modern workflow management system built for scalable and reproducible workflow execution; initially in the bioinformatics community and extended to other domains ¹. Nextflow utilises a domain-specific language extension of Groovy and, therefore, Java to assemble workflows out of processes. Parallelism is purely data-driven, and containerisation via docker or singularity achieve component reproducibility. Apache Ignite and Kubernetes support provides scalability and integrates with Amazon Web Services natively. Well-known open-source components form Nextflow, a modern workflow management system.

¹<https://github.com/nextflow-io/awesome-nextflow>

1.4.10 REANA

Computational workflows have found their way into a menagerie of scientific fields; Chen, Dallmeier-Tiessen, Dasler, *et al.* [47] outline the needs for a workflow management system in high-energy particle physics. Having open access to methods and data is not enough to make a dependable scientific claim, and hence workflow management systems somewhat tailored to individual fields are a necessary part of improving general scientific quality. To this end, Šimko, Heinrich, Hirvonsalo, *et al.* [106] present REANA, a workflow management and enactment system for re-usable analysis. Built with modern open-source tools such as Docker and Kubernetes allows REANA to leverage modern, industry-trusted computational tools in a scientific setting. While developed with high-energy particle physics in mind, REANA may find use in other scientific fields.

1.4.11 DALiuGE

DALiuGE is a prototype workflow management system built at the International Centre for Radio Astronomy Research (ICRAR) for deployment on the Square Kilometre Array. Coping with a previously unprecedented data-flow rate coupled with continuous scientific operation influenced Wu, Tobar, Vinsen, *et al.* [107] to adopt a data-flow orientation to workflow design. In DALiuGE, a collection of application and data ‘drops’ form workflows with links between them representing data-flow paths. Initially specified in a purely logical environment, DALiuGE translates high-abstraction logical components (including control flow structures) into physical drops with increasing detail.

DALiuGE fires execution of initial drops, but all subsequent communication occurs in an entirely decentralised manner between the drops themselves. DALiuGE scales workflows from individual machines during development up to extreme-scale in real-time and represents a frontier in computational workflow management.

1.4.12 Workflow Provenance

Computational workflows are ultimately a tool, facilitating science at a previously untenable scale and complexity, and as the scope, scale and reliance of workflows grew, so too did our needs to report on workflow execution, environment and results. Workflow provenance is, in essence, information about the source and derivation of workflow-generated data products and changes made to workflow descriptions themselves [108]. Provenance chiefly promises to improve the quality of published results as knowing more information about the derivation of final data products improves confidence in derived scientific conclusions. Provenance is also helpful in encouraging data exploration, further parallelising scientific analysis, and teaching sound practice [108]. Not all workflow management systems contain consideration for provenance information, and several open problems remain; however, developments in workflow management have improved the picture somewhat.

Interoperability between workflow management system is a long-standing goal in the field. Gil, Gonzalez-Calero, Kim, *et al.* [109] show the feasibility of specifying workflows using constraint programming to select fundamental workflow components dynamically. Kacsuk, Kiss, and Sipos [110] address workflow interoperability explicitly, developing a workflow management portal that enables workflow execution over multiple grid facilities utilising differing middle-ware.

However, while interoperable workflow execution is feasible, such workflows diminish their power in expressing an actual computation without a shared provenance environment. With the development of computational workflows as fundamentally important research artefacts [111], Missier, Belhajjame, and Cheney [112] develop the W3C Provenance model to provide a common language capturing workflow design and enactment. Nevertheless, widespread, practically useful workflow provenance remains challenging without implementation by widely used workflow management systems [113]. Pérez, Rubio, and Sáenz-Adán [114] provide a systematic review of workflow provenance systems, focusing on some of the systems we have introduced so far: specifically, Taverna, Kepler, and Pegasus. The development of FAIR data standards [42] hold relevance in workflow provenance for two reasons; workflow management systems can produce FAIR compliant data-products, and workflows can themselves be FAIR compliant data-products [115].

The standardisation, production and ultimate use of workflow provenance information make workflows more than simple tools to automate complex or repetitive computations, but a fundamentally integrated part of the scientific process [108].

1.4.13 The State of the Art

In only a few decades, computational workflows are no longer novel inventions but integral scientific tools. Atkinson, Gesing, Montagnat, *et al.* [44] present a whirlwind review of the rise of the scientific workflow management literature. Major expected trends moving forwards ultimately focus on developing management systems tailored to truly different scientific needs and increasing workflow dynamics as extreme-scale workflows simultaneously push data scale and computation complexity. Embracing modern design sensibilities, Dispel4Py [116] embeds workflow management as a highly abstracted Python library, a trend shared by the development of REANA, dependent on industry-developed open-source toolings.

A new emphasis on workflow provenance and workflow scales demands extending now mature workflow management systems such as Taverna [79] and Pegasus [92] with Ruiz, Garrido, Santander-Vela, *et al.* [117], providing exemplifying the power of embedding domain-specific knowledge into a well-known system as Taverna.

Classifying 15 well-known workflow management systems for extreme-scale workflows, Silva, Filgueira, Pietri, *et al.* [118] reveals marked differences between the systems introduced here. They point expressly to the need for future systems to become more data-driven rather than the traditional focus on computing tasks. To this end, Wu, Tobar, Vinsen, *et al.* [107] introduce DALiuGE, a bespoke data-driven workflow management system in development for the Square Kilometre Array (SKA), designed chiefly to cope with the enormous processing requirements of the SKA [44].

Moreover, Silva, Filgueira, Pietri, *et al.* [118] cite dynamic provenance capture as a necessary development if workflows are to scale larger and ultimately conquer more significant scientific problems. Gaignard, Montagnat, Gibaud, *et al.* [119] leverage semantic web technologies to produce more practically valuable workflow provenance data in the medical sciences, later improving this approach by developing a domain-specific ontology [120]. Community-driven efforts such as the WorkflowHub [121]² to build and maintain reproducible workflows and the associated tooling [122] facilitating their

²<https://workflowhub.eu/>

construction bodes well for the future of workflow-driven science.

In a world where transparent, trusted, and highly-scaled data processing is essential to conquering humanity’s most significant challenges [123], workflow management will continue to evolve to become more extensive yet more specific, more complex yet more trusted and ultimately extend the scope of science to previously untenable areas [124]. Our final introductory section traces the origin of blockchains from distributed computing. They are the primary source of technological inspiration in this thesis, although in a unique manner; they are, like workflows, distributed computations.

1.5 Blockchains

Our introduction’s final section traces the origins of blockchain technology from the state machine replication problem framing blockchains as a technology lying on one end of a spectrum of distributed computations. This section lays the groundwork to place distributed scientific workflows along this distributed computing spectrum, in turn borrowing the learnings from this vast swathe of literature when defining and implementing scientific reproducibility in computational workflows.

1.5.1 State Machine Replication

We begin our journey with a now-classic computer science problem; state-machine replication. State-machine replication is a formalisation of implementing a fault-tolerant service with replicated servers. A set of states, input values, output values, transition function between inputs and states, an output function between inputs and states to output values and a designated starting state fully describes a state-machine [125]. A client machine can submit any supported request to any replicate state machine with identical (or state-consistent) results in a correctly replicated system. Increasing the number of replicas increases the number of faults tolerated at the cost of increased complexity. State-machine replication is a fundamental problem in any distributed computing application requiring trust.

The Byzantine generals problem [126] occurs where system components fail and knowledge of these failures is itself imperfect. Byzantine faults represent the most challenging class of fault possible in a state-machine replication instance.

In any distributed computing application, solving the state-machine replication problem and knowing the Byzantine fault tolerance of a resulting system characterise the level of trust plausibly placed in the system [127].

1.5.2 Distributed Consensus

Solving the state-machine replication problem demands replicas to agree upon the machine-state in the face of Byzantine faults. Doing so deterministically in an asynchronous environment with arbitrary faults is famously impossible [128] but is practically solvable by a non-deterministic algorithm or in a synchronous environment. The inability to create perfectly replicated state machines results in many possible solutions and applications, all with unique compromises. The growth in Internet services ignited a strong interest in developing practically useful consensus schemes.

The first practical distributed consensus algorithm for synchronous systems is Lamport’s Paxos [129] which does not support Byzantine faults. Castro and Liskov [130] provide the first canonical

Byzantine fault-tolerant consensus algorithm. Gilbert and Lynch [131] formalise Brewer’s conjecture stating a web service cannot be consistent, available, and partition-tolerant simultaneously, which is a direct consequence of the impossibility of finding perfect consensus. Lamport [132] later provides the lower bounds on reaching asynchronous consensus, establishing special cases where specialised algorithms exhibit exceptional bounds.

The Internet’s construction, combined with sophisticated consensus schemes, facilitate some of the most impressive computing services known.

1.5.3 Distributed Computing Applications

In the age of growing Internet applications, consensus mechanisms play an essential role in their operation. The needs of each application, in turn, affect the exact nature of consensus implemented. Kemme and Alonso [133] provide a suite of protocols designed for database applications, specifically providing updates eagerly across all database replicas. These protocols achieve guaranteed consistency and avoid bottlenecks by making a tradeoff with performance and fault tolerance. The growth of Google’s web infrastructure makes a most inspiring example. The Chubby lock service [134] provides a global lock primitive for file resources in replicated data centres. Chubby uses consensus to make a single decision; electing a leader. This leader then coordinates data access by client machines with confidence, leveraging that a single user (Google) effectively owns all machines making fault-tolerance assumptions simpler. By targeting a long-term locking mechanism (days and hours, not minutes or seconds), Burrows [134] trades performance for availability and reliability, trading further with performance to achieve greater fault-tolerance. Focussing consensus on only a few critical decision allows Chang, Dean, Ghemawat, *et al.* [135] to implement a wide range of highly scalable applications with a wide range of performance requirements.

The maturity of consensus and data-replicated approaches [136] brought a wider variety of approaches to designing fault-tolerant distributed services. Junqueira, Reed, and Serafini [137] present Zab, an atomic broadcast algorithm aimed at crash-recovery and high-performance. Marandi, Primi, and Pedone [138] provide several additions to state-machine replication schemes emphasising performance in increasingly parallel environments. Kapritsos, Wang, Quema, *et al.* [139] provide a performant method to achieve state-machine replication in multi-core servers based on verifying non-deterministic execution order, accommodating changing hardware designs. Corbett, Dean, Epstein, *et al.* [140], powered by state-machine replication, provide Spanner, a globally distributed database scaling data access to millions of machines over hundreds of locations and trillions of data points. Distributed consensus mechanisms help implement the most data-intensive computing systems of the early 21st century.

1.5.4 Blockchains

What is a Blockchain?

All previously discussed distributed computing systems assume a single owner of all machines, a fact used to gain performance at the cost of fault tolerance. What if we instead consider a fully peer-to-peer distributed computing system expecting heterogenous machine ownership?

Nakamoto [141] solves the state-machine replication problem under this assumption using an entirely different consensus mechanism, creating a new technology and industry in the process. A blockchain functions as an immutable ledger recording transactions between potentially un-trusting peers. Every peer keeps a copy of the ledger, executes a consensus protocol to validate new entries, groups them into blocks and create a hash-chain over these blocks. Blockchains are similar to most other technological innovations, a selection of well-known components assembled in an inventive manner.

The proof of work consensus model pits machines against a known intractable problem of finding a hash with a set number of leading zeroes first introduced by Dwork and Naor [142]. Under a traditional distributed computing perspective, Bitcoins and blockchains merely implement the same fundamental problem trading increased fault tolerance for decreased performance.

The wide variety of consensus, incentive and application models provide an alternative but vibrant approach to distributed computing, a digital currency being the most straightforward possible application [143].

1.5.5 Iterated Blockchain Technologies

The invention of a vast peer-to-peer network based on highly distributed consensus spawned several alternatives, which vary in several key attributes. Presenting several notable developments highlight the interplay between classical consensus schemes and novel schemes built for vastly different operating conditions and, conversely, what thinking from blockchain technologies affect traditional distributed computing.

We group developments into three key attributes; transaction data, consensus model and access model (permissioned vs. permissionless).

Transaction Data

The information included in blockchain blocks effect for which applications a given ‘fabric’ is useful. In Bitcoin, these are strictly transactions, but this is not a strict requirement on blockchains in general. Sasson, Chiesa, Garman, *et al.* [144] introduce zero-cash, a Bitcoin alternative that embeds zero-knowledge proofs into blocks, providing a genuinely anonymous payment system.

The concept of embedding more than simple ledger transactions into blocks leads to the concept of smart-contracts [145]. These contracts are essentially small, deterministic scripts used to build autonomous logic into a blockchain network. Ethereum [146] is a blockchain alternative aimed at providing more complex contract logic than Bitcoin by implementing a virtual machine for deterministic execution across platforms and prices contract execution in ‘gas’ units. Smart contract move blockchain technologies closer to the traditional state-machine replication and thus introduces developments and challenges of their own accord.

Sergey and Hobor [147] argue that given the highly distributed nature of blockchain executions, smart contracts are subject to similar concerns of non-determinism to those of traditional concurrent computing. The survey of possible attacks on smart contracts assembled by Atzei, Bartoletti, and Cimoli [148] confirms these suspicions, and the infamous digital-autonomous-organisation attack (DAO) [149] resulting in \$50 million stolen cryptocurrency to provide a real-world example of such issues.

Meta-languages such as Hawk Kosba, Miller, Shi, *et al.* [150] introduce aim to alleviate these issues by providing a meta-language for smart contracts and further improving contract privacy by compiling to anonymous primitives such as zero-knowledge proofs.

Consensus Scheme

Alternate consensus schemes significantly affect a blockchain's properties, and it is here we most strongly see the interplay from traditional distributed systems thinking. The consensus mechanism used and critically what information reaches consensus are among the most critical factors.

Kwon [151] is the first to bring traditional Byzantine fault tolerance algorithms to blockchains. Ongaro and Ousterhout [152] observe that Paxos is notoriously tricky to understand and implement. Their alternative consensus algorithm, Raft, is more easily provable given understandability is a core design goal. Eyal, Gencer, Sirer, *et al.* [153] introduce a combination of proof of work and Byzantine fault tolerance. Leader selection utilises the traditional proof of work mechanism. The leader creates and simply broadcasts intermediate micro-blocks containing the actual transactions. Limiting what information requires consensus to transaction bundles significantly improves scalability while maintaining strong fault tolerance. Vukoli [154] discusses a comparison between proof of work (PoW) and Byzantine fault tolerance (BFT) based consensus models in blockchains.

Communication requirements of PoW scale well with the number of machines involved but offers poor distributed computing performance, whereas traditional BFT methods scale poorly concerning communication demands but offer superior performance. Other explored alternatives include:

- Improving PoW methods such as parallelising chain commits.
- Reducing the communication overheads of BFT methods.
- Creating mixed PoW plus BFT consensus models.

Ethereum, for instance, is moving to a proof of stake model, where unlike Bitcoin, which relies on raw computing power to reach consensus, the volume of currency actors stake on their proposed transactions controls consensus. The assumption that actors who hold the most currency are the least motivated to act maliciously is more in line with how traditional currency systems work. Liu, Viotti, Cachin, *et al.* [155] propose a hybrid cross-fault tolerance (XFT) scheme which models network errors and malicious node activities separately. A more realistic adversary model yields a more performance but equivalently secure consensus scheme.

Access Model

The amount of implicit trust placed in a blockchain's peers affects which consensus models are applicable, data throughput and security. A permission-less blockchain allows any peer to propose and validate transactions. Public access (counter-intuitively) implies minimal trust between peers by placing a high amount of trust in the consensus protocol and network itself. Bitcoin is the canonical example of a permission-less blockchain. Conversely, based on the observation that we often design systems with at least limited trust between participants, permissioned blockchains partition peers into groups of varying permission levels.

Permissioned blockchains limit vital responsibilities such as transaction commitment to a known set of peers. Stratifying responsibilities effectively simplifies the consensus process permitting simpler consensus schemes, significantly improving distributed computing power with known security properties.

Androulaki, Barger, Bortnikov, *et al.* [156] present a blockchain platform not tied to a particular crypto-currency or consensus model called Hyperledger Fabric. This system finally provides a scalable, secure and flexible, highly distributed computing platform combining decades of state-machine replication research with novel blockchain propensities.

1.5.6 Blockchain Applications

Aside from the canonical example application of a cryptocurrency, the promise of trusted distributed computing between untrusting peers creates a wealth of possible applications spanning a wide range of industries. An increased number of novel blockchain implementations allow for more complex applications. Di Francesco Maesa and Mori [157] summarise the latest and most successful fields where blockchains provide tangible benefits over traditional methods. Some fields are inherently more suitable to this paradigm than others. Despite a large volume of proposed ideas, uptake of various solutions and propositions is challenging to achieve [158], [159]. Significant proposed benefits to blockchain integration include:

- Auditability provided by a tamper-proof ledger
- Fundamentally distributed design
- Security by building trust between many parties

The inherent distribution of Internet of Things (IoT) devices makes blockchains an enticing tool to enhance security and trust in potentially vast networks of devices [160]. The primary concern for this field is performance; IoT devices are generally low powered (computationally), and hence limiting expensive cryptography routines, communication and storage requirements are a priority. Balancing performance concerns with improved security is a balancing act proving challenging to achieve [161].

Supply chains are a similarly distributed use-case where establishing trust between several participating companies valuable [162]. Tracing goods from production to consumption (and possibly disposal) is a primary goal. Blockchain technology is starting to see real-world use in wider-scale trials [163].

Moving to micro-grids for energy distribution promises to optimise energy production in a decentralised manner [164]. Auditing energy distribution systems is of particular priority making blockchains a potentially helpful tool. The ability to closely monitor power meters combined with more complex network optimisation schemes promises increased efficiency of these systems integral to modern society [165].

Integrating blockchains into science and business workflows has received some limited attention. Thus far efforts include tracking data ownership in cloud environments [166], distributed lab-work [167], business workflow management [168] and cloud data integrity management [169].

Given the difficulties faced in producing useful solutions and their wide use based on blockchains, it may be best to introduce this paradigm sparingly and perhaps in a more inventive manner. Intro-

ducing this exciting technology prepares us to use fundamental blockchain technologies to help enable reproducible science for the Square Kilometre Array.

Before embarking on the rest of this thesis, the following section outlines our overall approach to addressing reproducible science for the Square Kilometre Array.

1.6 Approach

We take a moment to summarise our introduction, outline this thesis' remainder, outline our problem and allude to a possible solution. We first introduce scientific reproducibility, a wide-reaching, somewhat existential issue in science. Scientific reproducibility concerns are field-dependent; however, a deluge of data and the prevalence of computing compound concerns. We then dove into the philosophy of science to provide a partial historical account of how science has developed over the last century, investigate why reproducibility is desirable or an identified remedy, and find motivation for what a reproducibility mechanism should practically achieve. A philosophical motivation ensures our proposed solution integrates with contemporary science at a most fundamental level.

We direct our efforts ultimately at the Square Kilometre Array, one of the most collaborative and massive scientific endeavours faced with an order of magnitude data-processing challenge. Science with the SKA is critically bottlenecked by its computing resources yet observational at its core and increasingly automated.

We subsequently frame computational workflows as a contemporary method to orchestrate such computation, highlighting several notable workflow management systems in the process. We summarise our problem as: The SKA presents an opportunity to iterate upon what large-scale data processing and scientific best-practice entail. We propose a mechanism asserting workflow qualities and properties that will provide an open, certifiable, and testable method to enable reproducible computational workflows and reproducible science for the SKA.

Reviewing the derivation and state of the art of blockchain technologies hoping their distributed computing origins and promises of open trust will find a home in computational workflow science. While it is clear that blockchains will not provide an immediate solution directly, in this thesis, we show how repurposing blockchain primitives complements and extends current approaches to reproducing computational workflows.

We organise the remainder of this thesis as follows: Chapter 2 presents an arbitrary scale data-processing workflow model and map seven well known scientific workflow management systems to it. Mapping existing workflow management systems to a standard formulation ensures our approach and discussions apply to arbitrary computational workflows since addressing all systems is prohibitively time-consuming. We subsequently formulate seven reproducibility tenets as testable assertions placed on our data-processing workflow model in Chapter 3. A test-driven approach to workflow reproducibility complements developments from the philosophy of science and, more practically, current workflow provenance capture. Chapter 4 thoroughly explains our blockchain-inspired approach to implementing our defined tenets, including a reference implementation in the DALiuGE workflow management system. To foreshadow, we build a block directed-acyclic-graph (DAG) over workflow descriptions and executions, extracting critical provenance information to provide a workflow signature for the desired reproducibility tenet. This chapter outlines what information, in particular, is of interest and

details the building of the BlockDAG. The blockchain inspiration arises from framing computational workflows as automated, complicated distributed computations. While we apply our implementation to DALiuGE specifically, our mappings make application to other workflow management systems possible. Ultimately, this approach yields an amortised constant-time method to test various reproducibility tenets at an arbitrary computing scale autonomously. Penultimately, Chapter 5 validates our approach by applying each tenet to a low-pass filter workflow, implemented in both DALiuGE and natively without a workflow management system according to our model defined in Chapter 2. Finally, Chapter 6 concludes our discussion with a summary and suggestions for future inquiry.

Chapter 2

Scale-Agnostic Scientific Data Processing

This chapter introduces a novel, scale-agnostic model for scientific data-processing and maps seven well-known computational workflow management systems to it. Workflow interoperability is a challenging problem [104], [170], and while systems to describe generic computational workflows exist [171], [172], our model is different in two key dimensions. First, we aim for a model potentially embedded within several workflow systems or implemented natively; hence, we define our model in Unified Modelling Language (UML). While further model formalisation into ontologies for generalised scientific processes [173] allows accurate descriptions of data provenance [124], further formalisation is beyond the scope of this thesis, whose focus lies on direct software implementation and testing of reproducibility concerns. Second, our model explicitly separates logical workflow design from physical implementation, thus permitting scale-agnosticism. In later chapters, we see this separation from logical and physical workflow descriptions permits a highly expressive approach to reproducibility at arbitrary data scales. Finally, the ability to reason separately about logical workflow design, physical implementation and ultimately, execution captures the entire workflow lifecycle, including published workflows. Later, we find that reproducibility assertions defined on our model are by extension applicable to any mapped workflow management system making these assertions widely commensurate.

2.1 Workflow Model

Figure 2.1 presents our scientific data-processing workflow model. Our model possesses several notable traits. The UML definition is abstract but allows for direct software implementation. The alternating of logical and physical tasks with data artefacts ensures our model captures all interim data. Moreover, interleaving data and computing components make every individual data-compute-data triplet a complete workflow, a base-case definition making our model scale-agnostic. A decentralised execution pattern possible only with a fundamentally scale-agnostic definition alleviates the need for centralised coordination. We later see that fine-grained introspection is not always possible in all workflow management systems. Moreover, the distinction between logical and physical tasks is a crucial model feature. Different implementations for physical tasks can perform the same logical task. Several physical tasks executing in parallel can implement a single logical task, providing the basis for our scale-agnostic

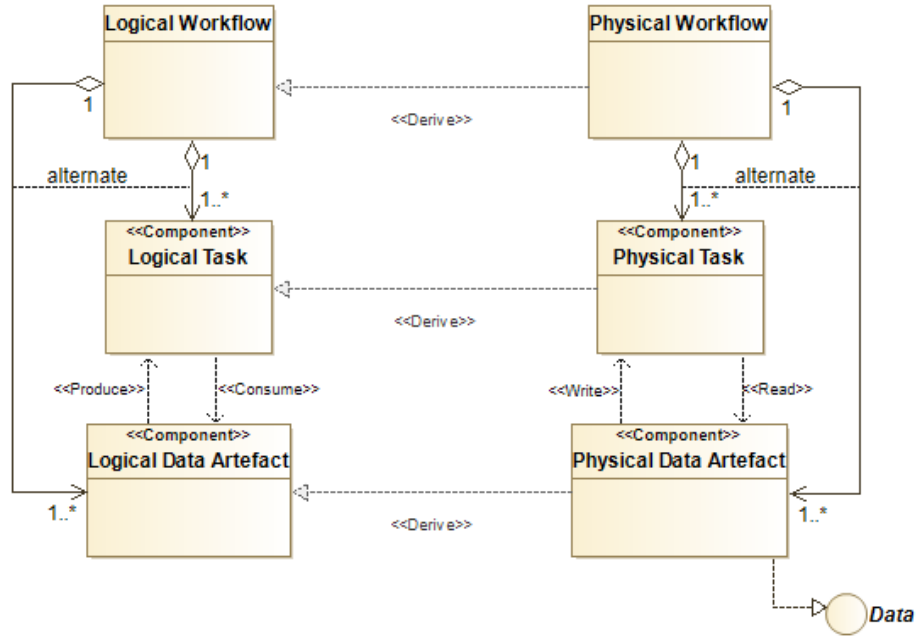


Figure 2.1: A UML diagram depicting our arbitrary-scale workflow definition.

A **component** is an atomic digital entity.

A **physical task** is a single computation characterised to the level of accuracy we choose.

A **logical task** is a single abstract computation characterised by only essential details like programming language or algorithm.

One or more physical tasks realise a each logical task. A **physical data artefact** is a datastore like a file, distributed file system or database.

A **logical data artefact** is a logical data resource characterised by data-type alone. Many physical data artefacts can provide equivalent data access, and a single logical data artefact may encompass many individual physical data artefacts at runtime.

Scientific information comprises initial and terminal logical and related physical data-artefacts and effectively expresses the information used in making a scientific claim.

A **workflow** is a collection of components and data artefacts with an imposed ordered structure.

This ordered structure requires components and data artefacts must appear in alternating order; that is, another component cannot precede a component.

A **physical workflow** is a workflow comprised of physical tasks and physical data artefacts.

A **logical workflow** is a workflow comprised of logical tasks and logical data artefacts.

claim. Finally, complex logical tasks can comprise a sub-workflow of logical tasks, providing more nuanced workflow expression. Our model implicitly includes control structures (loops and if statements, for example) since these are fundamentally logical tasks.

This model aims to capture the execution pattern of many different workflow management systems but not be the basis for a new implementation. We believe this model is simple yet expressive enough to capture a wide range of scientific computing tasks and captures a wide range of scientific workflow descriptions.

2.2 Workflow Management System Mappings

Reasoning about any possible workflow and workflow management system is likely impossible or, at the very least, challenging [110]. Mapping several existing workflow management systems to our model allows reasoning on our workflow model to cascade onto mapped workflow management systems. Here we demonstrate our model's flexibility and highlight some inherent similarities and differences between mapped systems. Moreover, any additional system mappings may benefit from original reasoning applied to the model, making the ability for some systems to express reproducibility concerns in a commensurate manner concrete.

For each mapped workflow management system, we briefly remind readers of its history and usage, motivate its selection for analysis, provide and discuss our mapping and briefly discuss any prior reproducibility efforts concerning that particular workflow management system. The review Silva, Filgueira, Pietri, *et al.* [118] provide motivates our selection, and we believe our selection provides a good cross-section of approaches and use-cases for scientific data-processing workflows.

2.2.1 Taverna

History and usage Taverna was one of the first scientific workflow management systems conceived and primarily orchestrates series of web services as data-processing workflows [118]. Taverna was the first system to consider packaging workflows as separate research objects [44]. Recent additions and consistent development have allowed Taverna to facilitate more complex workflows logics and parallel execution.

Selection for analysis Taverna has been widely used in the bioinformatics community and heavily influenced subsequent generations of scientific workflow management systems. The web-service oriented approach is additionally a good robustness test for our workflow model.

Mapping to workflow model Taverna expresses workflows as graphs of computational components, which are typically web-services [117]. As such, mapping Taverna workflows to our model is non-trivial but still possible and contained in Table 2.1. Notably, the second generation of Taverna separates the data and workflow execution spaces via a data-manager, which assigns data items unique identifiers [79]. The separation of data references and items best captures our model's difference between logical and physical data items. However, the process of querying the data-manager for a given data item may be missing in Taverna workflow executions mapped to our model.

Table 2.1: Taverna to workflow model element mapping.

Workflow Model Element	Taverna Element
Component	
Logical Task	Activity
Physical Task	Processor
Logical Data Artefact	Data Link
Physical Data Artefact	Data Item
Logical Workflow	Workflow Object (Model)
Physical Workflow	Executable Workflow Graph

Existing reproducibility efforts Taverna considered workflow provenance from the very beginning [75] and is an essential element. Taverna now supports the W3C Provenance standard [80] but does not consider reproducibility and re-usability explicitly [111] opting instead to trust the well-defined workflow model implicitly.

2.2.2 Askalon

History and usage ASKALON [85] is a workflow management environment for grid-computing. AGWL, an XML schema for workflows [86] is the language of ASKALON workflows. Teuta exposes a graphical and UML-like interface to AGWL workflows [87]. The intended principal benefit is the separation of operating concerns from workflow design, leading to the exploration of novel scheduling and runtime environments.

Selection for analysis ASKALON is one of the earliest examples of a scientific workflow management system and pioneered developing a resilient scheduling system in complex parameter spaces [44], [118]. The well-defined workflow language, focus on scalability and separation of concerns between workflow design and enactment make ASKALON a natural choice for mapping to our workflow model.

Mapping to workflow model The XML schema ASKALON implements [88] has an understandably heavy focus on expressing parallelism. The ASKALON enactment engine instantiates ‘reified’ workflows on a grid-service [174]. Table 2.2 provides a mapping of similar elements in ASKALON to our workflow model. Notably, ASKALON does not define logical data artefacts as objects but as ports, a recurring

Table 2.2: ASKALON to workflow model element mapping.

Workflow Model Element	ASKALON Element
Component	Action
Logical Task	Activity Type
Physical Task	Activity Deployment
Logical Data Artefact	Data Ports
Physical Data Artefact	System Files
Logical Workflow	Compound Activity
Physical Workflow	Activity Set

motif in several mappings. ASKALON’s fine-grained approach to parallelism will make separating

reproducibility concerns challenging. Moreover, the schema-based approach is rigorous, as evidenced by the inability to express data-artefacts as discrete entities.

Existing reproducibility efforts ASKALON does not implement reproducibility standards explicitly. Extensive use of the Java runtime environment will provide consistent behaviours at a low level, at least for the workflow manager. ASKALON explores workflow management as a science and comes from a ‘pre-crisis’ era concerning scientific reproducibility. A rigorous workflow schema guarantees a level of correctness and uniqueness for ASKALON workflows.

2.2.3 Pegasus

History and usage Pegasus [90], [92] executes workflows in distributed environments by mapping an application workflow onto available resources. Pegasus focuses on scalability and reliability with widespread use in various science applications [118].

Pegasus contains various intelligent features to improve performance and reliability; for instance, non-shared filesystems automatically include staging in/out jobs where needed.

Selection for analysis Pegasus has demonstrated scale and performance in a wide variety of scientific endeavours [44] makes it a canonical choice as a specifically distributed workflow management system. Moreover, Pegasus is task-agnostic, allowing workflow construction for a wide array of scientific application.

Mapping to workflow model Directed acyclic graphs in XML (DAX) is the main language of Pegasus workflows. Additionally, data objects are discrete entities, making mapping pegasus workflows to our model relatively straightforward. Table 2.3 contains our mapping of similar Pegasus elements to our workflow model.

Table 2.3: Pegasus to workflow model element mapping.

Workflow Model Element	Pegasus Element
Component	
Logical Task	Abstract Task Nodes
Physical Task	Executable Task Nodes
Logical Data Artefact	Edges / Logical Filenames
Physical Data Artefact	System Files
Logical Workflow	Abstract workflow
Physical Workflow	Executable workflow

Existing reproducibility efforts The sheer volume of intelligent systems built into Pegasus makes precise workflow recomputation an enormous difficulty in highly distributed and dynamic environments. However, Pegasus includes some reproducible components as performance improvements. Pegasus maintains a replica catalogue of existing data artefacts, removing needless recomputations and instead reuses data Deelman, Vahi, Juve, *et al.* [92]. Garijo, Villanueva-Rosales, and Kauppinen [175] manage

to embed provenance semantics into workflow descriptions and executions, in effect capturing the scientific context of workflow execution.

2.2.4 Kepler

History and usage Kepler [94] is among the first workflow management systems conceived and outlines several requirements for successful workflow management systems. Kepler inherits a unique director and actor model from the underlying Ptolemy II Java library and finds use in a wide range of scientific domains [118]. The actor-director paradigm offers a unique approach to workflow enactment. Without changing the logic of a Kepler workflow, changing director components change actor execution patterns.

Selection for analysis Kepler’s unique director and actor model, comparison to business workflow systems, subsequent involvement in the development of the Open Provenance Model [176] and W3C Provenance standard [112] and use in a wide range of scientific domains make Kepler a significant development in scientific workflow management systems and hence an essential inclusion [44].

Mapping to workflow model Mapping Kepler to our workflow model is non-trivial given the actor-director system. We consider the choice of actors and directors logical components, but upon enactment, directors instance the final workflow tasks. Like ASKALON, data ports connect computing tasks and are the closest analogous components to logical data artefacts. Table 2.4 contains our mapping of similar elements in Kepler to our workflow model.

Table 2.4: Kepler to workflow model element mapping.

Workflow Model Element	Kepler Element
Component	Component
Logical Task	Actor/Director
Physical Task	Executable
Logical Data Artefact	Data Ports
Physical Data Artefact	System Files
Logical Workflow	Model
Physical Workflow	Workflow

Existing reproducibility efforts Data provenance is a core requirement of the Kepler system. Kepler generates provenance data in XML format for export, and Altintas, Barney, and Jaeger-Frank [95] provides a good explanation of this system. Kepler also contains a very forward-thinking ‘smart-rerun’ concept similar to Make build systems where workflow components are only re-executed if an implementation or dataset change exists.

2.2.5 Nextflow

History and usage Nextflow [105] is a modern workflow management system built from open-source components. Bioinformaticians use Nextflow where high-performance is necessary; the vast array of supported executors make Nextflow pipelines widely useful.

Selection for analysis Nextflow represents the state of the art in Bioinformatics, a prominent first-adopting field of workflow technologies. Moreover, the assembly from industry-tested open-source elements is a departure from the bespoke engineering of original workflow management systems.

Mapping to workflow model Nextflow employs a domain-specific language based on processes and files assembled with various operators. Table 2.5 contains our mapping of similar Nextflow elements to our workflow model.

Table 2.5: REANA to workflow model element mapping.

Workflow Model Element	Nextflow Element
Component	
Logical Task	Process
Physical Task	Process
Logical Data Artefact	Filename
Physical Data Artefact	File / Input Stream
Logical Workflow	Pipeline
Physical Workflow	Workflow

Existing reproducibility efforts Di Tommaso, Chatzou, Floden, *et al.* [105] built Nextflow with reproducibility in mind, containerising Nextflow pipelines is extremely simple in Nextflow and requires a single command-line option. The success of the nf-co.re repository efforts [177]¹ illustrates the benefits of building reproducibility into workflow management systems directly. Beyond containerisation and the existence of workflows themselves, Nextflow offers several introspection options to identify individual workflow executions and workflow versions.

2.2.6 REANA

History and usage REANA [106] is a relatively new workflow management system built from well-known open-source industry tools like, Kubernetes [178], HTCondor [179] and Docker [180] built primarily for the high-energy particle physics community. REANA expresses workflows as YAML files in a relatively simple workflow description but is continuously adding new features.

Selection for analysis REANA is a new system designed to integrate modern tools and a specific focus on reproducibility for data-intensive sciences. REANA represents the current state-of-the-art concerning what a scientific workflow system should encompass and makes an excellent candidate for consideration.

Mapping to workflow model REANA accepts two directed acyclic graph workflow models; the Common Workflow Language [181] and Yadage [172]. As such, mapping REANA workflows to our model is relatively straightforward. Table 2.6 contains our mapping of similar elements in REANA to our workflow model.

¹<https://nf-co.re>

Table 2.6: REANA to workflow model element mapping.

Workflow Model Element	REANA Element
Component	
Logical Task	Container Description
Physical Task	Container Instance
Logical Data Artefact	Filename
Physical Data Artefact	File contents
Logical Workflow	Workflow Description
Physical Workflow	Workflow

Existing reproducibility efforts Scientific reproducibility is a core goal in REANA’s design and implementation. Chen, Dallmeier-Tiessen, Dasler, *et al.* [47] base their rationale on the definitions laid out by Barba [33] but applied explicitly to the high-energy particle physics community. This framework is of particular interest as it lays out core motivations behind each reproducibility tenet (rerun, repeat, reproduce, replicate and reuse). Nevertheless, REANA only models workflows as a static directed-acyclic graph of parameterised components, a straightforward and practically functional approach. Implementing our workflow model in REANA is simultaneously trivial and challenging, owing to REANA’s inherent simplicity. Specifically, the inability to explicit separate a logical workflow description from a physical workflow makes comparisons between workflows a mostly human endeavour.

2.2.7 DALiuGE

History and usage DALiuGE is a computational workflow manager built by the Data Intensive Astronomy team at the International Centre for Radio Astronomy Research (ICRAR) for the SKA, its precursors, and other pathfinder systems [107]. DALiuGE models computations first as graphs comprised of vertices corresponding to computational tasks and data artefacts and edges corresponding to the data’s natural flow. At their highest abstraction layer, graphs may contain control structures but are ultimately translated down to a directed-acyclic graph before instantiation on available computing resources. This paradigm focuses on the most critical challenge posed by SKA science processing, dataflow. Abstracting important information about a computation in a workflow permits manipulation and use adjacent to merely running a computation.

Selection for analysis The SKA, at peak capacity, will need to process an enormous volume of data in real-time without scientist intervention. Ensuring this automatic processing is of sufficient scientific quality to permit subsequent scientific analysis is key to the SKA becoming a successful scientific venture. DALiuGE aims to solve a future data-processing challenge that presents an opportunity to build scientific reproducibility into the system during development rather than after deployment and hence is a system we focus on intensely in this thesis.

Mapping to workflow model DALiuGE specifies a workflow at two distinct layers; logical and physical. The logical layer captures workflow design independent of available compute resources. A graph template provides the most abstract representation of a data-processing workflow and may possess several parameters. A fully parameterised logical graph template becomes a logical graph and

specifies the prospective science. Scientists design logical graph templates well in advance, reusing well-known components without exposing exacting technical details. DALiuGE allows astronomers to design and execute highly performant computational workflows at scale without demanding expert software engineering knowledge.

The physical layer captures the execution of a workflow cognisant of available resources. A translator with machine-specific knowledge unrolls logical graph tasks into their atomic, directed-acyclic form. A physical graph template represents the actual volume of computation and order of tasks that needs subsequent mapping to actual machines. A fully specified physical graph adds exact machine information, runtime data and is the closest abstraction to the performed computation.

Table 2.7 summarises the mapping of our generic data-processing workflow model and DALiuGE.

Table 2.7: DALiuGE to workflow model element mapping.

Workflow Model Element	DALiuGE Element
Component	Component
Logical Task	Logical Component
Physical Task	Application DROP
Logical Data Artefact	Logical Data Component
Physical Data Artefact	Data DROP
Logical Workflow	Logical Graph
Physical Workflow	Physical Graph

Existing reproducibility efforts DALiuGE contains some reproducibility functionality. In addition to fail-over processing, DALiuGE contains the capacity to replicate physical computing tasks designated as ‘precious’ on separate machines to mitigate hardware failure.

Finally, this thesis’ principal technical objective is to design and implement a system facilitating comprehensive scientific reproducibility in DALiuGE and is thus a core goal of DALiuGE’s design.

2.3 Discussion

We here discuss the differences in our workflow mappings. The scientific target domain of a workflow management system strongly influences its design. Mapping several such systems to a single workflow model is, therefore, a naturally tricky task. However, asserting any scientific reproducibility constraints between workflow management systems is made easier if there is a common framework to apply such constraints to. We identify three characteristics of the workflow management systems discussed pertinent to our workflow model mapping:

- Modelling computations as individual tasks or hosted services.
- Addressing data elements as workflow objects or not.
- Separating logical workflow components from the physical components performing the computation.

Table 2.8 summarises these differences between the seven mapped workflow management systems discussed in this chapter. The service-based approach of Taverna, ASKALON and Kepler makes

Table 2.8: Summary of workflow management system design characteristics

	Service-Based (S) Task-Based (T)	Data Artefacts Modelled Separately	Logical / Physical Abstraction
Taverna	S	✓	✓
ASKALON	S		✓
Pegasus	T	✓	✓
Kepler	S		✓
Nextflow	T	✓	
REANA	T	✓	
DALiuGE	T	✓	✓
Native	N/A		

mapping physical tasks problematic as they are only implicitly present when a service processes a data element. The task-oriented approach of Pegasus, Nextflow, REANA and DALiuGE make mapping logical and physical tasks easier to identify. Taverna, Pegasus, Nextflow, REANA and DALiuGE all allow data-store modelling and included as separate workflow entities making assertions on data integrity easier to interpret. REANA is the only system mapped that maintains a skinny layer of logical abstraction from an actual workflow. The lack of logical abstraction makes REANA simpler and thus more robust but also inherently less flexible.

Finally, we include an entry for a native computation included as a baseline reference. Our model must be workflow management system agnostic, including the lack of a management system. The lack of imposed structure makes assumptions about data modelling and logical abstraction challenging to make and a service or task-based approach impossible. However, our UML composed model makes a native implementation of a compliant workflow possible.

2.4 Conclusion

This chapter introduces a data-processing computational workflow model specified in UML. This model essentially alternates computing and data components and abstracts logical workflow components from their physical implementation. The alternation of components captures all data processing, and the two layers of abstraction make our model scale-agnostic. We subsequently map seven workflow management systems to our model that range in management paradigms, age and target scientific discipline. We find that service-based workflow management systems make abstracting physical computing tasks somewhat ephemeral, and modelling data artefacts as separate entities is a more recent trend that aids in making assertions on data. We also discuss applying our model to a native computation devoid of a workflow management system. Finally, it is possible to apply our scale-agnostic data processing workflow model to vastly different workflow management systems reasonably, providing a platform to apply scientific reproducibility constraints widely and commensurately.

Chapter 3

Reproducible Scientific Data Processing

Embedding computing into scientific processes such that the science is reproducible is a well-discussed but unsolved problem. One could demand that a reproduced computation requires precisely the same software, data and hardware, but in practice, not only is this unreasonable but arguably unhelpful. A single computation is multi-faceted orchestration of logic and technology, and there are scientific motivations for reproducing various subsets of a computation. Applying scientific reproducibility concerns to computational workflows has been previously discussed, and here we continue the discussion and hopefully provide some semblance of a solution. This chapter formally defines several well-discussed reproducibility tenets for our scale-agnostic data-processing workflow model as a collection of testable assertions. Moreover, we frame our tenets within the context of the philosophy of science, motivating our approach from scientific rather than technical first principles. Defining reproducibility tenets on our previously introduced workflow model makes our definitions actionable and widely applicable to any mapped workflow management system. Following a general discussion of our definitions and their motivation, for each tenet we provide:

- An unambiguous definition.
- A justification for each definition. In general, each tenet is commensurate with pre-existing interpretations, but we point out discrepancies where applicable.
- A few motivating example scenarios, providing practical motivation behind our definition.

Our formal definition of several reproducibility tenets for a workflow model paves the way for transparent integration into several existing workflow systems with minimal end-user intervention. We use the popular terminology to rerun, repeat, recompute, reproduce and replicate computational workflows as these terms capture intuitive concepts around scientific reproducibility [32], [33], [47], [113]. Enabling formal verification of scientific reproducibility tenets across several workflow management systems embeds computational workflow as an increasingly important component of high-quality data-intensive science.

3.1 Philosophical Motivation

Regardless of which philosophy of science one subscribes to, a few truths are abundantly clear: Framing science as a human endeavour of individuals collaborating to accomplish tasks is appropriate, and experimentation is a hallmark of the scientific process. Computing is also a very, perhaps counter-intuitively, human endeavour as we essentially build on others' technical decisions to perform novel computations. Immediately, we see a difference between abstract computations we intend to carry out, and their physical implementation. This difference will play a crucial part in integrating computing into science; in principle, it should never matter how we performed a computation. Processed data is of equal importance where a similar nuance is required; asserting reproduction is down to matching a series of bytes removes all scientific context. The development and popularisation of computational workflows is an effort to separate abstract computational logic from implementation. Doing so makes reasoning about increasingly complex and extensive computations easier.

Previous attempts to integrate computing into the scientific reproducibility discussion focus on assuring future software behaviours, creating a 'forward-acting' approach to enable reproducible science. Such efforts range from asserting standards on individual pieces of code [32], enacting well-defined data management protocols [42], [114], containerising whole workflows [106], 'top-down' measures on behalf of journal publishers [182], and conference organisers [183] and careful provenance production and collection [124]. While certainly practical, this thesis proposes that forward-facing measures insufficiently integrate computing into science; they are ultimately empirical and require an inductive step to evaluate their efficacy.

This thesis chiefly argues for the need for 'backwards-facing' measures to test if subsequent workflow executions for scientific credibility. Such a system, to be scientific, needs to demarcate:

- Implementation from abstract computation
- Data and method reproduction
- Workflow similarities agnostic of the workflow management system
- And any combination of these traits therein.

This approach borrows the Popperian approach of conjecture and refutation to complement pre-existing approaches encouraging workflow reproducibility but applies regardless of technical implementation. Table 3.1 summarises our seven tenet definitions as component invariants defined on our previously discussed workflow model (Chapter 2). In effect, we reduce the discussion of reproducibility to a more tractable decision surrounding what elements are scientifically essential and why reproduce them?

Table 3.1: Reproducibility tenets summarised as workflow component invariants.

	Logical Tasks	Physical Tasks	Data Artefacts	Scientific Information
Rerun	✓			
Repeat	✓	✓		
Recompute		✓		
Reproduce				✓
Replicate (Scientific)	✓			✓
Replicate (Computational)		✓	✓	✓
Replicate (Total)	✓	✓	✓	✓

3.2 Rerun

Definition - Rerun A workflow *reruns* another if they execute exactly the same logical workflow; that is, their logical components and dependencies match.

Justification Following from Chen, Dallmeier-Tiessen, Dasler, *et al.* [47], who motivate rerunning to show used tools are robust, we propose that rerunning in the context of computational data-processing workflows is analogous to using an identical algorithm independent of its implementation. Asserting consistency only on the logical components in a workflow allows physical component changes. This definition, in turn, allows changes in computing scale, parallelism patterns, hardware, data-encoding or software implementation while still constituting a rerun. If all physical details are kept identical, a successful rerun builds trust in the underlying tooling used to deploy a workflow, and a failed rerun demonstrates system failure.

It may, at first, appear that rerunning is a rather weak claim since rerunning encompasses relatively few workflow components. Consider the following scenarios:

- Developing a complex workflow on local machines for deployment on vast distributed resources, asserting a rerun between a small local and scaled-up execution demonstrates that the scaling-up process does not change the underlying workflow logic.
- Changing physical task implementation (changing an underlying numerical library or language runtime version, for instance) is commonplace during development. Testing that such changes do not affect the logical workflow in effect tests whether the logical workflow sufficiently describes the intended workflow logic. The ability to make such claims encourages publishing and reuse of logical workflow designs specifically.
- Conversely, when extending or modifying a workflow, failing to demonstrate a rerun asserts the existence of a meaningful design change.
- For many data ingest workflows, rerunning may be the only possible comparison between executions as resource volumes and raw data-artefacts are, by design, different.

- When using new, or public, compute resources, repeatedly rerunning workflows builds confidence in the underlying resources.

Rerunning maximises logical workflow reuse, a desirable property when developing and scaling-up data-processing computational workflows.

3.3 Repeat

Definition - Repeat A workflow *repeats* another if they execute the same logical workflow and a principally identical physical workflow; their logical components and dependencies and physical tasks match. The specificity of a ‘principally identical’ physical workflow allows for successful recovery from failed physical tasks (due to hardware or operating failure).

Justification Repetition is a relatively well-defined concept, and we generally follow intuition. Chen, Dallmeier-Tiessen, Dasler, *et al.* [47] motivate repetition to build statistical power behind results. Cohen-Boulakia, Belhajjame, Collin, *et al.* [113] suggest repeated scientific computational workflows match all components (workflow specification, data input and runtime environment) by analogy to laboratory work. Our definition mostly follows these interpretations but provides a concrete definition. Repeating a scientific computational workflow asserts reuse of the logical and physical workflows, which, if successful, allows bundling of data-artefacts in concert. Our interpretation is somewhat analogous to what Goodman, Fanelli, and Ioannidis [14] term methods reproducibility. Importantly, we leave the amount of information clarifying physical tasks as an environment and problem-dependent decision but insist that hardware details are not pertinent for repetition. Consider the following scenarios:

- Data will change when investigating naturally stochastic phenomena. Direct assertions placed on data-artefacts may be inappropriate, but asserting repeat trials through testing workflow repetitions builds statistical power in the aggregated data products.
- In scenarios where sufficient compute resources to process an entire dataset are not available, batch processing large datasets is necessary. Successful repetition tests assert identical processing of each batch.
- In extreme-scale scientific workflows, hardware failure is inevitable. Such a scenario replaces an identical physical task and does not alter the resulting physical workflow.
- Surprising unsuccessful repetition attempts may reveal that the workflow in question is fundamentally non-deterministic, which may or may not be essential but is now known and captured.

Repetition is a fastidious test of method reuse, which helps build defensible statistical power in data-artefacts.

3.4 Recompute

Definition - Recompute A workflow *recomputes* another if they execute the same physical workflow; that is, their physical tasks and dependencies match precisely.

Justification Recomputation is effectively a stricter interpretation of computational workflow repetition. The core difference to repetition is that recomputation requires identical physical tasks, identical hardware and is not fault-tolerant. Workflow provenance, taken to an albeit extreme conclusion motivates this definition; the amount of effort required to recompute a given workflow exactly is immense and is likely impossible without total hardware control. The concept of repetition Benureau and Rougier [32] present is somewhat similar as they call principally for the removal of non-determinism. In distributed computational workflows, the distribution itself can cause remarkably complex numerical non-determinism that is painstaking to correct [184], and as such, recomputation is a seldom justified endeavour. Consider the following scenarios:

- When moving from a single machine to a distributed computing resource, verifying recomputations helps assert the absence of non-determinism on account of the distributed workload.
- Upon publication or discovery of incredibly novel or critical results, verified recomputation assures that the results are intentional.
- When establishing the use of a new workflow management system, verified recomputation of known workloads builds confidence in the quality of these new tools.
- If computer hardware development or exploitation (such as the infamous meltdown [185] and spectre [186] exploits) is the science in question, recomputation becomes a critical goal towards demonstrating a novel contribution.

Successful recomputations ultimately build trust in the implementation and workflow execution infrastructure.

3.5 Reproduce

Definition - Reproduce A workflow *reproduces* another if their scientific information match.

Justification Scientific reproduction is a widely discussed topic, and our definition does not conflict with existing literature. Our definition of reproduction demarcates the net data flow a computational data-processing workflow consumes and produces. Generally, reproducibility proper refers to reproducing experimental results [1], a task that is extremely difficult in lab-based science [18]. Benureau and Rougier [32] define a reproducible code as a trivially executable one that produces published results. Our definition focuses solely on data in a similar interpretation to Chen, Dallmeier-Tiessen, Dasler, *et al.* [47] and by extension Barba [33]. Cohen-Boulakia, Belhajjame, Collin, *et al.* [113] take a more comprehensive definition that two reproducing workflows must reach the same conclusion. Gundersen [48] provides a detailed comparison of many computational reproducibility definitions, highlighting dissonance between various definitions.

However, defining reproducibility in terms of invariants and variants between workflow executions is useful. Our definition focuses on data-artefacts specifically to provide a more actionable approach given the enormous amount of freedom researchers have in workflow design can affect scientific outcomes despite similar intermediate data-products [37]. Our approach is also compatible with reproducibility approaches involving constrained computing environments [45] since we place no constraint on logical or physical task components. Focussing on data reproduction specifically places our definition more firmly in the broader framework of tenets, focusing on different scientific elements. Testing for reproducibility is helpful in the following broad scenarios:

- Scientists iterating on a workflow design can change logical or physical tasks with more efficient implementations or significantly altered paradigms and test that the new workflow reproduces the previous iteration. Beaulieu-Jones and Greene [45] present a similar approach natively using continuous integration methods.
- Independent verification of scientific results is possible if a third party can successfully reproduce the published workflow results.

Our definition of reproduction facilitates workflow development.

3.6 Replicate

Replicability in science is a substantially debated concept with many different interpretations [1], [48]. As such, we incorporate replication into our framework as combinations of our previous tenets, creating three different but related standards which all constrain workflow tasks and data-artefacts.

Definition - Scientifically Replicate A workflow *scientifically replicates* another if they *rerun* and *reproduce* each other.

Justification Our definition asserts that a principally identical computational workflow produces the same terminal data-artefacts, in effect verifying that a logical workflow can produce a given set of results. This interpretation of replication is reminiscent of conceptual replication discussed by Fidler and Wilcox [1] and of Benureau and Rougier [32] who state a code is replicable if its textual specification provides enough information to reproduce its results. Cohen-Boulakia, Belhajjame, Collin, *et al.* [113] hold the same approximate definition over workflow components but differ over an entire workflow in that a replicated workflow should support the same conclusion. Testing for scientific replication is helpful in the following scenarios:

- When scaling up a workflow to exploit greater parallelism, in principle, this should only change physical workflow components, and one can test against known datasets for correctness.
- Changing physical component implementations for performance gains without alteration to known behaviour.
- Comparing workflows between management systems is expressly possible, either for development purposes or independent verification of workflow logic.

Scientific replication permits the reuse of workflow logic without compromising known results.

Definition - Computationally Replicate A workflow *computationally replicates* another if they *recompute* and *reproduce* each other and all data-artefacts match.

Justification Our definition asserts that a precisely identical computational workflow produces the same data-artefacts at every stage, which is the strictest interpretation of a replication down to specific hardware details. This replication interpretation is similar to the concept of direct replication that Fidler and Wilcox [1] discuss and is closely related to workflow provenance discussed widely in computational workflow literature [114]. The information required to replicate a workflow execution computationally is immense as it demonstrates that a specific workflow implementation on specific hardware yields a given set of results, building confidence in workflow implementation but not necessarily the underpinning science; it would be challenging for an independent team to replicate a complex workflow computationally. As such, computationally replicating workflow execution may be helpful in the following scenarios:

- Performing science on computer hardware specifically; testing the performance characteristics of new circuitry or toolchains.

- Chasing bizarre workflow execution issues, in effect replicating errors and failure to characterise its actual cause.

Computational replication represents the upper limit of control possible over a workflow execution.

Definition - Totally Replicate A workflow *replicates* another if they *repeat* and *reproduce* each other and all data-artefacts match.

Justification Our definition asserts that a practically identical computational workflow produces the same data-artefacts, including interim data, and are identical down to software, but not hardware, details. This interpretation of replication is consistent with that of Chen, Dallmeier-Tiessen, Dasler, *et al.* [47] who motivate replication tests as a method to certify a workflow as a ‘gold-standard’ and is somewhat analogous to what Goodman, Fanelli, and Ioannidis [14] term results reproducibility. A total replication permits independent verification of results by a third party and balances scientific and computational replication. A total replication balances scientific and computational replication and is applicable in the following scenarios:

- A total replication allows independent result verification by a third party since the only allowed changes are trivial hardware differences.
- Scientists who are looking to establish their workflow execution as deterministic.
- When replicating old workflows on new hardware, a total replication is the most robust assertion available.

Total replication balances scientific and computational replication and establishes confidence in workflow results.

3.7 Discussion

Our reproducibility tenets specify which elements of computational workflow executions concerning a generic workflow model should be kept identical. This approach has the practical benefits of being concretely defined and actionable, programmatically testable without human intervention and applicable to arbitrary workflow management systems. Inspiration from the philosophical concept of conjecture and refutation ensures our approach is technology agnostic and complementary to modern approaches for workflow reproducibility. Aside from providing tangible execution proof via provenance data, rerunning, repeating, reproducing and replicating computational workflows matters most when trying to do so. As such, instead of trying to guarantee future reproducibility via enforcing a strict standard, our definitions reframe this problem to finding ways of demonstrating and comparing performed work. Moreover, automatic testing provides the practical benefit of improving trust in scientific tools, in turn highlighting more nuanced and complex sociological issues in science [10].

There are also several scenarios where failing to achieve a reproducibility tenet reveals valuable information. If expectedly unsuccessful, a workflow execution is decidedly novel in some attribute, whether it be the workflow logic, execution environment, data or physical task implementation. Unexpected failure hints at the insufficient characterisation of a workflow execution or phenomena, providing actionable methods to improve the confidence in workflow-driven science. One may address inherently stochastic phenomena by either building a statistic model out of a series of workflow executions or, perhaps a more rigorous option, is adding additional logical tasks summarising or characterising stochastic results. Making computational workflow reproducibility dynamic and actionable over arbitrary workflow management systems opens an array of use-cases for reproducibility information rather than being simply a chore.

3.8 Conclusion

This chapter motivated a test-driven approach to computational workflow reproducibility to define seven reproducibility tenets. These tenets are testable assertions placed on our previously discussed workflow model and provided several examples where this approach betters workflow-driven science. By reconciling the quest for computational reproducibility with the philosophy of science, the resulting approach is concretely defined and therefore actionable without human intervention.

Chapter 4

Methods

This chapter presents our blockchain-inspired approach to enact our reproducibility tenets in the DALiuGE workflow management system. This implementation realises the mapping we make in Chapter 2, and while we describe our implementation concerning a specific workflow management system, the techniques and algorithms described may apply to any mapped workflow system. To do so, we first describe the cryptographic primitives needed in our approach. Then, we provide a broad system description explaining how workflow verification works over a workflow’s lifetime. Following, we provide algorithmic details of each component and an account of what information is pertinent for each tenet for several workflow components. Finally, we conclude with some complexity analysis. Our reference implementation is on GitHub ¹.

4.1 Cryptographic Primitives

Blockchains, as discussed previously, are, like any other technical innovation, a clever assembly of pre-existing technologies. We describe the data-structures blockchains employ in a bottom-up fashion, extending the basic concepts to a block directed-acyclic graph (DAG).

4.1.1 Hashing

Hashing is a method of applying a cryptographic hash function to arbitrarily large pieces of data to produce a relatively unique fixed-size output [187]. Hashing algorithms allow individuals to independently take input data, hash the data and derive the same result. If two hashes are identical, it is overwhelmingly likely that the data inputs are identical. More specifically, a proper hashing function has:

- Preimage resistance - The function is one-way, meaning it is computationally infeasible to find the input that produced a given digest.
- Second preimage resistance - Given a specific input, finding a second input that hashes to the same output requires an exhaustive search.

¹<https://github.com/pritchardn/daliuge>

- Collision resistance - Finding two inputs that hash to the same output requires an exhaustive search.

There are several commonly used hashing functions with standardised implementations. A more complex hashing function is more computationally intensive to execute but provides greater security. The advent of hardware-based encryption furthers standardisation; a common choice is SHA-256. The 256 specifies a hash output of 256 bits. Recently, however, many are moving to the SHA-3 family of algorithms. Simpler algorithms such as MD5 are no longer cryptographically secure² but still function to provide unique hash values.

4.1.2 Merkle Trees

Merkle trees are binary search trees with additional properties. Leaves contain a data element and a hash of this data, and internal nodes contain a hash of the left and right sub-tree hashes. Merkle [188] designed and patented Merkle trees in 1979. Leaves store data, combined into pairs and hashed recursively, terminating with a single root. Merkle trees allow fast comparison of structured data since the same roots imply identical contents. Upon failure, a recursive search can find mismatched leaves in $\Theta(\log n)$ time. Figure 4.1 depicts a Merkle tree comprised of four data elements. Merkle trees find

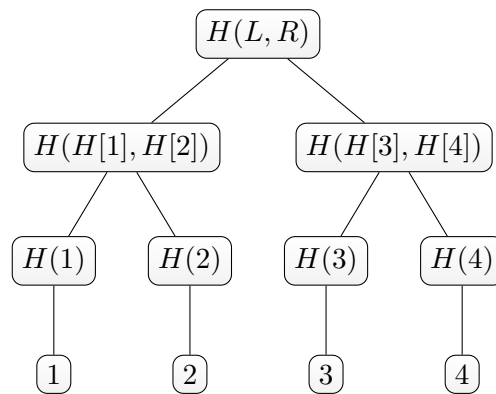


Figure 4.1: A simple Merkle tree comprised of four data elements. $H(x)$ is a hashing function for arbitrary data.

use in distributed file systems to ensure consistency between copies. A naïve approach would send complete file copies; a better approach would send a hash of the contents. If the two hash values match, there is no need to send any more data. If the two hash values do not match, sending the hashes of the left and right children will test if the two sub-trees match. This process recurses until identifying the specific file parts that differ. A Merkle tree provides an asymptotically optimal method to compare contents sending a minimal number of hashes across the network on average. Merkle trees find use in:

- Git - Stores data files in Merkle Trees, allowing changes on one machine to propagate to all other participating users quickly.
- Blockchains - A block's transaction list become leaves in a Merkle tree, allowing blocks to process multiple transactions at once.

²<https://www.kb.cert.org/vuls/id/836068>

- Database replication - Similar to Git, changes across multiple database copies are efficiently propagated instead of comparing raw and often sizeable database components.

4.1.3 Merkle Directed Acyclic Graphs

Merkle Directed Acyclic Graphs (DAG) are a conceptually simple extension to Merkle trees: nodes can have multiple children, and all nodes can store data, not just the leaf nodes. These two changes allow Merkle DAGs to store significantly more arbitrary data but requires more sophisticated comparison algorithms. The comparison routine now grows (computationally) with the branching factor of the graph, which is highly dependent on the data; consider a chain of nodes, for instance, as a worst-case. Merkle DAGs are a powerful abstraction tool and is the base data-structure of the Interplanetary File System (IPFS) [189].

4.1.4 Block Chains

Blockchains, the data structure and not the entire network are a particular type of Merkle DAG with no branching and all nodes containing new data. A block typically contains:

- An ID value
- Creation timestamp
- Data payload (transaction list or otherwise)
- Merkle tree root of the data payload
- Previous block's hash
- Current block hash (comprised of all data listed above)

Including the parent block's hash, any retroactive change to a previous block drastically changes all subsequent blocks' hashes. Therefore, it is possible to verify the integrity of all data in a blockchain by verifying the hashes of all previous blocks. Blockchain networks employ various consensus mechanisms to agree upon what the next block contains, usually adding some information to the block in the process (like a nonce value in Bitcoin's proof of work, for instance).

4.1.5 Block DAGs

A directed-acyclic graph of blocks is a BlockDAGs, a MerkleDAG with arbitrary connections and all nodes containing data. Terminal leaf nodes in a BlockDAG rely on intermediate node hashes. By collecting the leaf node hashes into a Merkle Tree, the root will contain a 'signature' for the entire BlockDAG. BlockDAGs are the foundational data structure of our approach to enabling workflow execution verification. Our approach does not implement a blockchain but a different piece of technology that shares the same core data-structures and algorithms; hashing algorithms, Merkle trees and blocks.

4.2 System Overview

Our solution is conceptually simple, build a BlockDAG over a DALiuGE workflow. The information stored in each block by each component changes for each reproducibility tenet. We collect the hashes of leaf nodes, insert those into a Merkle tree, and take the root hash as a signature for that workflow for a particular tenet. We present a straightforward hypothetical example workflow to explain how the hash-graph generation relates to DALiuGE’s workflow enactment.

DALiuGE workflows exist in several stages up to and including execution. Workflows begin as logical templates comprised of exclusively logical components and control structures. Scientists design logical graph templates several months in advance and for extensive re-use. Figure 4.2 presents the logical graph template for a simple averaging workflow and its associated hash-graph. Values read from an initial file are scattered across multiple machines for averaging; we then collect these intermediate averages and save them to a final output data-artefact. We maintain the original graph structure, and each component generates a hash-value related to its parents’ hash and its specification by inserting all information into a Merkle tree and taking its root. A logical graph is a logical template with specified options (number of input channels, for instance). Logical graph specification occurs on the order of a month before execution. Figure 4.3 presents the logical graph for our example. The structure has not changed, but drops contain more information (indicated by the ‘fields’ entry), which changes their resulting hash values.

DALiuGE then unrolls logical graphs into physical graph templates, resolving control structures and presenting the net computation required. This process is in practice unique for each logical graph but in principal changes depending on the translator implementation. By unrolling, these physical graph templates is now a directed acyclic graph. Unrolling is a complicated process, with several possible algorithmic approaches occurring around a week before execution. Figure 4.4 presents the physical graph template for our example, where DALiuGE scatters the data across two branches. Each component contains yet more specifying information as each represents an instance of a processing component or data-artefact, but principally identical components will hold identical hash-values. DALiuGE typically partitions an unrolled graph immediately unless machine resource information is unknown. Alternatively, partitioning an unrolled graph using several different algorithms yields a variety of parallelism profiles and is machine dependent. Figure 4.5 presents our example physical graph where all components (now drops) now contain their exact machine information. Principally identical drops are now different because of the graph partitioning, and the resulting hash-values reflect this.

DALiuGE finally instances the first drops in a physical graph but thenceforth drops communicate to each other, driving graph execution in an entirely decentralised manner. Figure 4.6 presents the our example’s final runtime graph where drops contain a status value and data-hash (if applicable to the required reproducibility tenet and drop type).

Notably, while Figures 4.2 to 4.6 present the hash graphs for each workflow abstraction in isolation, in practice we hash these levels together to tie the entire workflow lifecycle into a single signature. Figure 4.7 presents an entire hash-graph where parent hashes refer to predecessors in the current graph and equivalent drop in previous abstractions. This simple example has a single leaf node and minimal

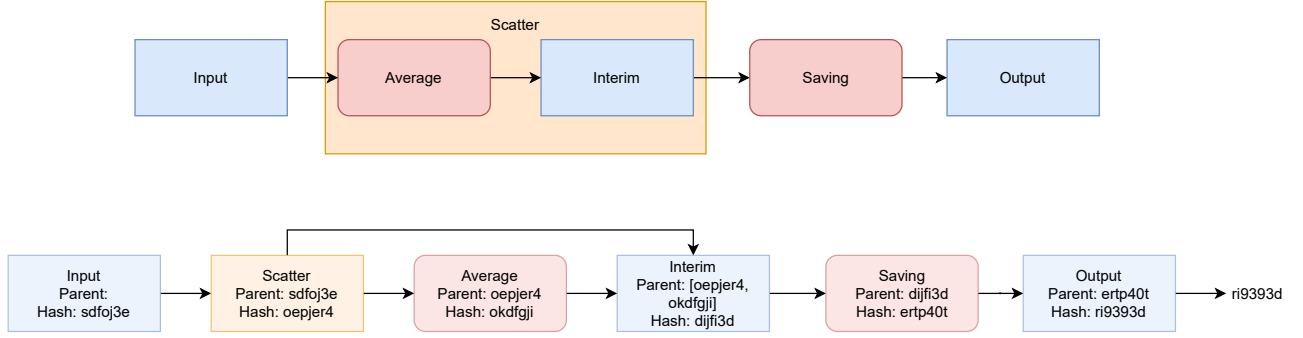


Figure 4.2: Example Logical Graph Template and associated hash-graph.

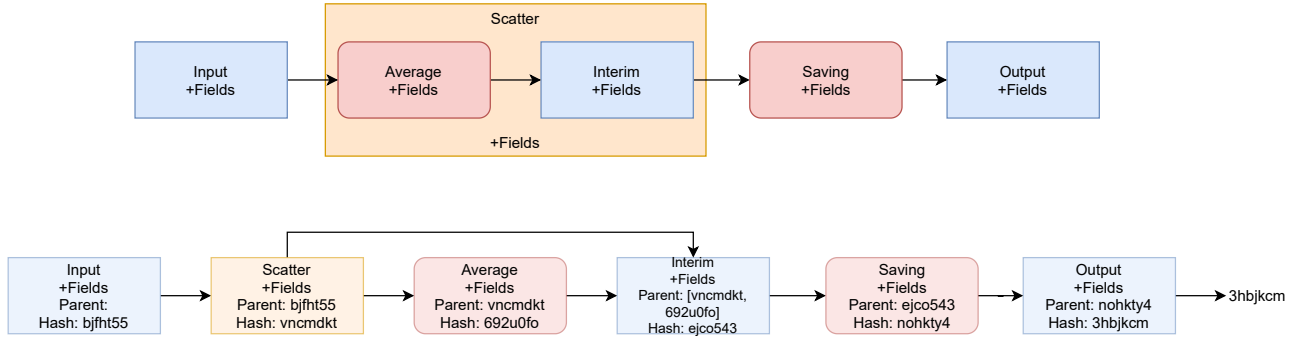


Figure 4.3: Example Logical Graph and associated hash-graph.

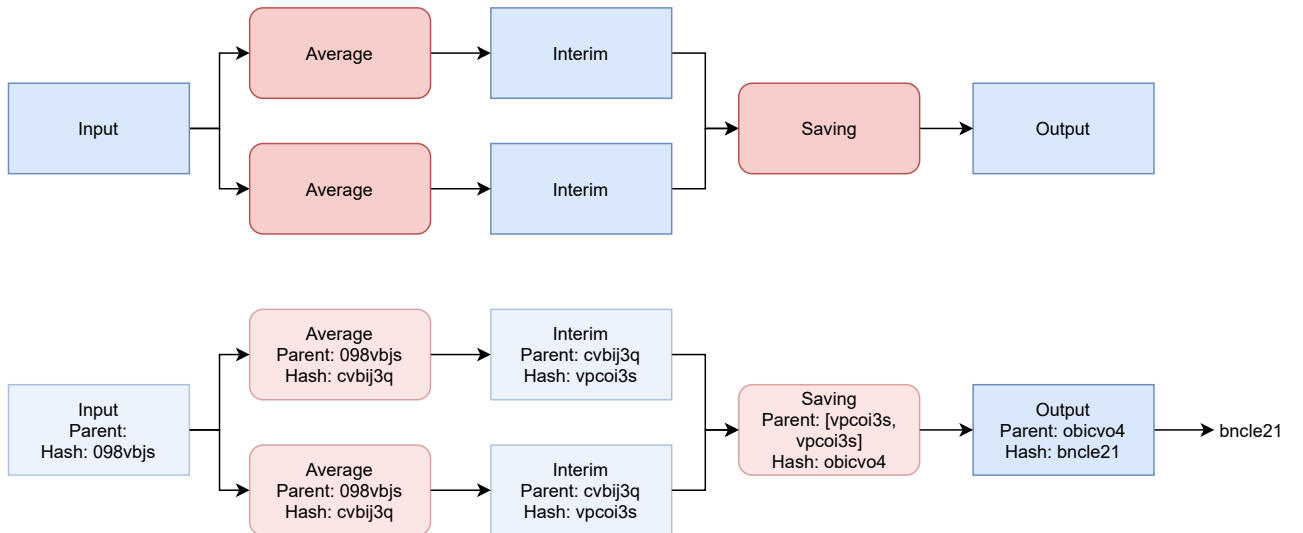


Figure 4.4: Example Physical Graph Template and associated hash-graph.

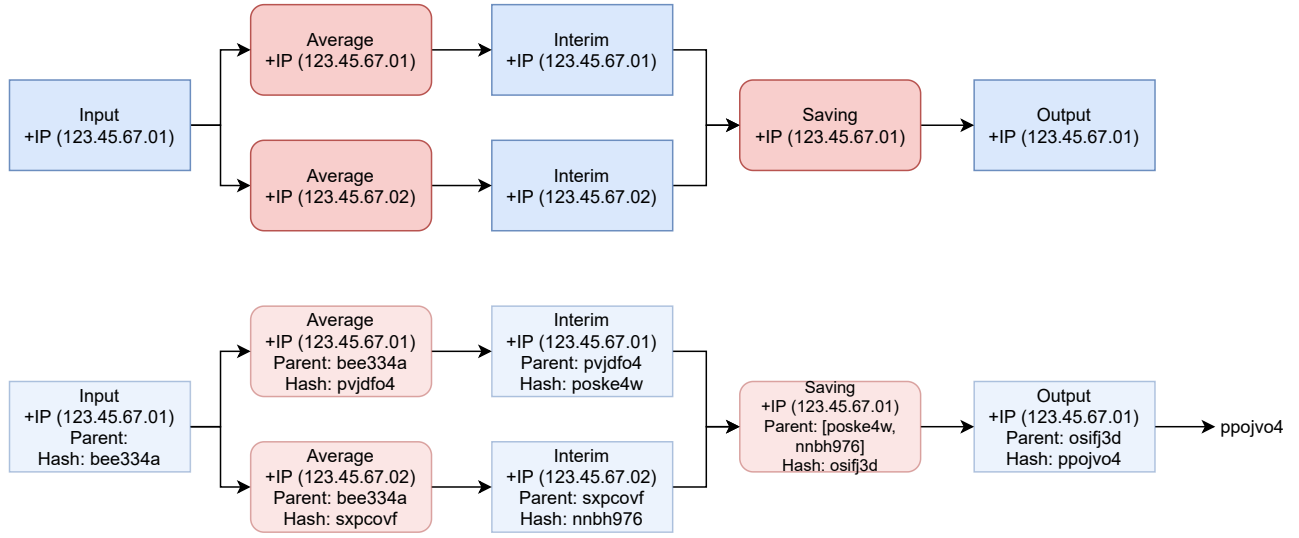


Figure 4.5: Example Physical Graph and associated hash-graph.

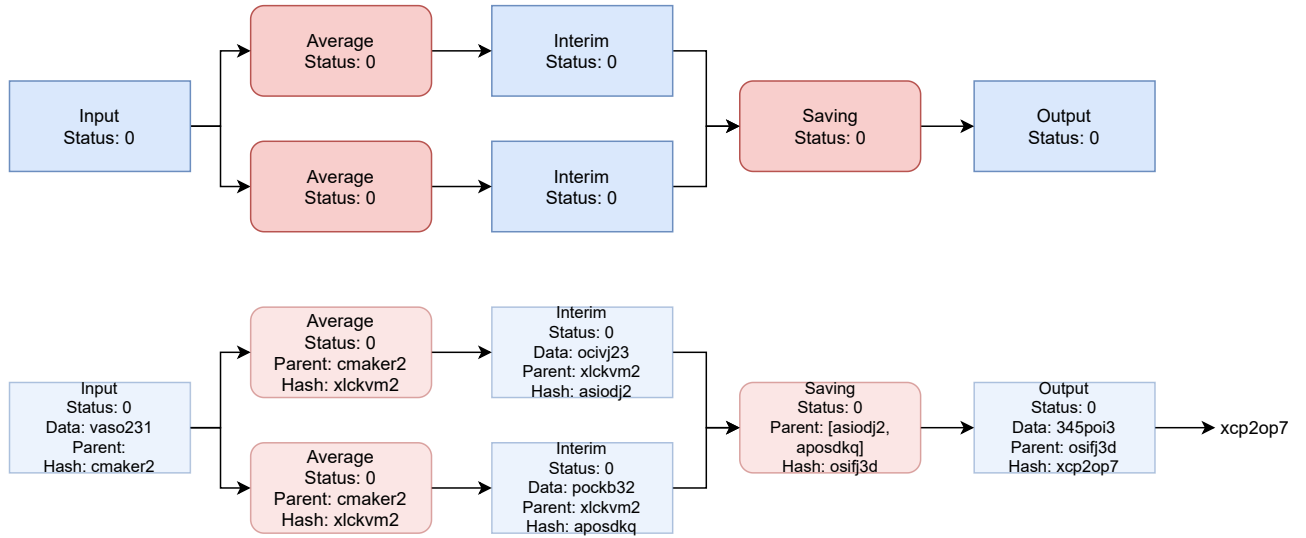


Figure 4.6: Example Runtime Graph and associated hash-graph.

parallelism and is relatively straightforward; it should be unsurprising that visualising more complex examples becomes arduous.

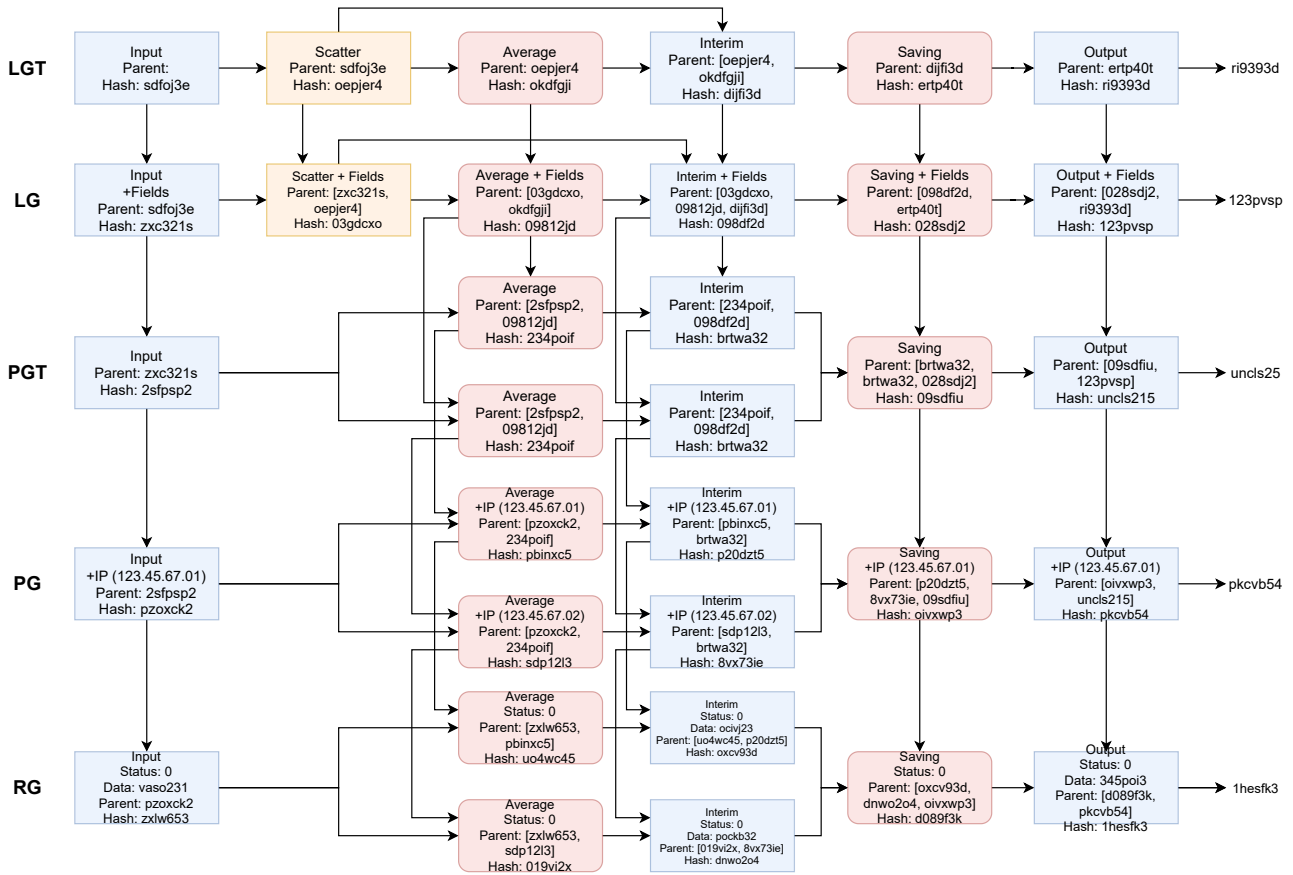


Figure 4.7: Example combined hash-graph.

We aim to enforce reproducibility tenets at any stage in this cycle, and as such, we intercept all DALiuGE translation routines to construct our BlockDAGs. Figure 4.8 presents a UML sequence diagram of this life-cycle, including interactions with our reproducibility code base. DALiuGE centralises all operations until physical graph execution. Constructing the runtime BlockDAG is significantly more complicated but conceptually simple. Tiers of node managers instance physical drops. Each drop produces and sends its reproducibility information back up the manager tiers for insertion into the final BlockDAG. Concerning our mapping, building a BlockDAG over workflow components at each level of abstraction captures the relationships between logical tasks, physical tasks and data-artefacts and separates logical and physical workflow signatures. Depending on the reproducibility tenet in question, we add or remove pertinent information to these BlockDAGs, potentially omitting some components entirely. Building the BlockDAG at each layer does not increase each operation's asymptotic complexity, and the total amortised constant-time complexity of imposing any reproducibility check is a requirement for ska-scale deployment. Testing for tenet compliance with a previous workflow execution at any abstraction level then takes $\Theta(1)$ time; the complexity of comparing two signatures.

Each mapped component must offer its provenance information and expose its relationships to other components. The following subsections, in turn, discuss constructing the BlockDAG and what information DALiuGE keeps for each reproducibility tenet for each component (drop) type at each

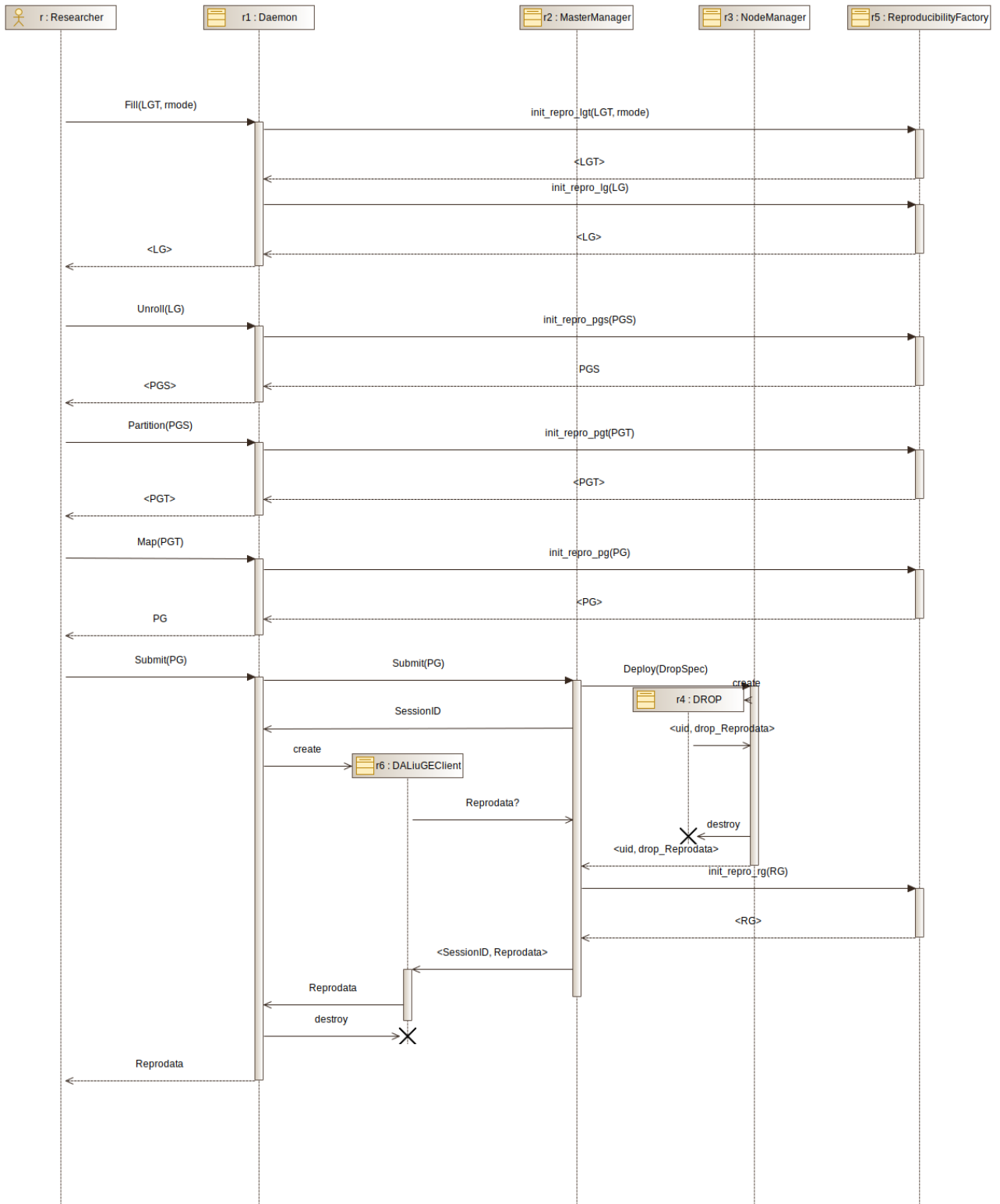


Figure 4.8: UML Sequence diagram specifying the complete life-cycle of a DALiuGE workflow with reproducibility concerns

level of workflow abstraction.

4.3 BlockDAG Construction

Building a BlockDAG over a workflow description is a graph-traversal problem. We need to visit every component precisely once and in topological order. We employ Kahn’s algorithm for topological sorting [190]. While not DAGs, logical graphs (templates) have at least one starting component and for signature generation, ignoring cycles is a necessary step. For a graph $G(V, E)$, Kahn’s algorithm iteratively finds all vertices with no incoming edges, adds them to a queue, removes all outgoing edges from the graph, effectively maintaining a frontier of roots. Algorithm 1 contains pseudocode for our procedure, which in addition to graph-traversal, builds each component’s (drop’s) block and appends parent hashes to child hashes, building a BlockDAG. The input G is a collection of component (drop) descriptions and their connectivity. After execution, each component description will hold an additional reproducibility block at the specified abstraction level (GA), conforming to reproducibility tenet R . Notably, behaviour changes depending on the requested reproducibility tenet; when replicating, we propagate data-artefact hashes through all task components, and if reproducing specifically, we only include the initial and terminal data-artefacts. We insert leaf hashes into the final Merkle tree in alphanumeric order for determinism. This operation in isolation has a runtime of $\mathcal{O}(V + E)$, processing each edge and vertex precisely once.

The block building procedure, outlined in Algorithm 2 inserts parent hashes and relevant information into a Merkle tree. By including hashes from higher levels of abstraction, the final signature depends on all previous steps. This operation runs in $\mathcal{O}(n \log n)$ time and space where n is the number of data elements included in a block.

In the worst-case, a block depends on all previous blocks directly incurring a runtime of $\mathcal{O}(V \log V)$. The total runtime of BlockDAG generation is $\mathcal{O}(V(V \log V) + E) = \mathcal{O}(V^2)$. A worst-case scenario would be a fully connected graph, an overwhelmingly unlikely scenario for a workflow management system. Thus, we characterise BlockDAG generation runtime as $\mathcal{O}(V(D \log D) + E)$ where D is the average vertex degree. We expect the average vertex degree will be much smaller than the number of vertices.

Building a BlockDAG over workflow components is a core element of our solution. We now review what information various component and drop types insert into their blocks for each reproducibility tenet.

4.4 Component and Drop Fields

This section summarises what information each drop type stores in its block for each DALiuGE workflow abstraction level and each reproducibility tenet. For brevity, we aggregate drops into three categories, application, data and control. Describing stored information for each reproducibility tenet is critical as this information encodes the mapping from DALiuGE workflow components to our model components.

Input: Graph G , Reproducibility Mode R , Graph Abstraction GA

Output: BlockDAG signature

$q \leftarrow \text{Queue}()$

$\text{Visited} \leftarrow []$

$\text{Leaves} \leftarrow []$

for *Component* $c \in G$ **do**

if $G.c.in = \emptyset$ **then**

$q.append(c)$

if $G.c.out = \emptyset$ **then**

$\text{Leaves.append}(c)$

end

while $!q.empty()$ **do**

 Component $curr \leftarrow q.pop()$

$\text{build_block}(curr, GA)$

$\text{Visited.append}(curr)$

for *Component* $child \in G.curr.out$ **do**

$G.child.in.remove(curr)$

$\text{Parents} \leftarrow []$

if $R = RP \text{ or } RPL(S \text{ or } C \text{ or } T)$ **then**

if $G.curr.type = \text{Data}$ and $G.curr.in = \emptyset$ or $G.curr.out = \emptyset$ **then**

$\text{Parents.append}(curr.hash)$

else

$\text{Parents.append}(curr.parents.hash)$

end

if $R \neq RP$ **then**

$\text{Parents.append}(curr.hash)$

$G.child.Parents.append(\text{Parents})$ **if** $G.child.in = \emptyset$ **then**

$q.append(child)$

end

$\text{LeafHashes} \leftarrow []$

for *Leaf* $L \in \text{Sorted}(\text{Leaves})$ **do**

$\text{LeafHashes.append}(L.hash)$

end

end

return $\text{MerkleTree}(\text{LeafHashes}).root$

Algorithm 1: BlockDAG Construction via Kahn’s algorithm. Reproducibility Mode R is one of Rerun (RR), Repeat (RT), Recompute (RC), Reproduce (RP), Replicate Scientific ($RPLS$), Computational ($RPLC$) or Total ($RPLT$). Graph Abstraction GA is one of Logical Graph Template (LGT), Logical Graph (LG), Physical Graph Template (PGT), Physical Graph Stencil (PGS), Physical Graph (PG), Runtime Graph (RG).

Input: Component c , Graph Abstraction GA
Result: Component c with filled hash
 $\text{block} \leftarrow []$
 $\text{hashset} \leftarrow \text{set}(c.\text{parents})$
for $\text{hash } h \in \text{hashset}$ **do**
 $\text{block.append}(h)$
end
if $GA \neq LGT$ **then**
 $\text{block.append}(c.(GA - 1).\text{hash})$
 $\text{block.append}(\text{generate_block_data}(c, R, GA))$
 $c.GA.\text{hash} \leftarrow \text{MerkleTree}(\text{block}).\text{root}$
return c

Algorithm 2: Block Construction. Reproducibility Mode R is one of Rerun (RR), Repeat (RT), Recompute (RC), Reproduce (RP), Replicate Scientific ($RPLS$), Computational ($RPLC$) or Total ($RPLT$). Graph Abstraction GA is one of Logical Graph Template (LGT), Logical Graph (LG), Physical Graph Template (PGT), Physical Graph Stencil (PGS), Physical Graph (PG), Runtime Graph (RG).

4.4.1 Rerun

Our definition requires two workflow executions are *reruns* they must match their logical workflow and logical components. Our mapping to DALiuGE implies the logical components and data drops, and, ultimately, logical graph specifications must match.

As such, when asserting a rerun in DALiuGE, we capture meaningful information at the LGT and LG layers with successively less information captured and only a successful execution flag at the runtime level. Physical drop execution acts as execution proof for their logical counterparts.

Table 4.1 contains a broad summary of block information at each workflow abstraction and each type of drop. A component’s logical ‘type’ is a generic label (bash script or python component, for example). A component’s in and out ports refers to how many input and output connections it has. A component’s PGS ‘type’ is potentially more specific, referring to the component instance used (a particular version of Python, for example), and ‘name’ refers to the exact component used (a particular Python script, for example). Finally, the ‘status’ field at the physical level is the execution flag used to ascertain a drop’s successful or unsuccessful execution. It is clear that we only store broad, logical details of all drop descriptions. We effectively treat application drop execution as proof for the corresponding logical component.

Table 4.1: Information stored to assert workflow reruns for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
	Type					
Application	InPorts OutPorts	-	Type Name	-	-	Status
Data	“	“	Type Storage-Name	“	“	Status
Control (misc)	“	“	Type	“	“	-

4.4.2 Repeat

Our definition requires, for two workflow executions to be *repetitions*, they must precisely match on logical workflow and logical components. Our mapping to DALiuGE implies the logical components and data drops, and, ultimately, logical graph specifications must exactly match.

DALiuGE demonstrates repetition by storing more logical details than what rerunning captures, Table 4.2 reflects this change. Importantly and recurringly, the fields reference applies to several variables changing with each specific application type (a command for a bash script component or image tag for a docker component, for example).

Table 4.2: Information stored to assert workflow repetitions for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type InPorts OutPorts	Num-CPU Fields	Type Name	-	-	Status
Data	“	Data-Volume Fields	Type Storage-Name	“	“	Status
Control (misc)	“	Fields	Type	“	“	-

4.4.3 Recompute

Our definition requires, for two workflow executions to be *recomputations*, they must precisely match on physical tasks and their dependencies. By our mapping to DALiuGE, this implies the application components must *exactly* match.

Table 4.3 summarises what information we capture to assert a workflow recomputation. There are a few fields of note. ‘Rank’ refers to the logical ordering of decomposed sub-tasks. Analogous to the rank of MPI sub-processes. The ‘Node’ and ‘Island’ fields at the PGT and PG levels store which machine executes that drop as an integer reference and IP-address, respectively. The ‘Trace’ information captures executed code but depends on the particular drop’s implementation and language.

Table 4.3: Information stored to assert workflow recomputations for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type InPorts OutPorts	Num-CPU Fields	Type Rank Name	Node Island	Node-IP Island-IP	Status Trace
Data	“	Data-Volume FileNames Fields	Type Rank Storage-Name	“	“	Status
Control (misc)	“	Fields	Type Rank	“	“	-

4.4.4 Reproduce

Our definition requires, for two workflow executions to be *reproductions*, they must match on logical and physical data drops. Our mapping to DALiuGE implies logical data components, and data drops must match in both specification and content between two workflow executions.

Table 4.4 summarises what information we track to assert workflow reproductions in DALiuGE. The focus is on the runtime ‘Data-Summary’ of all data drops. This summary could be a hash of all file contents in simple cases or a pre-existing summary mechanism built into the component itself (consider a FITS file [191], commonly used in Astronomy). Moreover, we build a reproducibility block-dag by only including data drops.

Table 4.4: Information stored to assert workflow reproductions for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type	-	-	-	-	-
Data	“	“	Type Storage-Name	“	“	Status Data-Summary
Control (misc)	“	“	-	“	“	-

4.4.5 Replicate

By our definition and mapping to DALiuGE, to show two workflow executions are *scientific replicas* they must be *reruns* and *repetitions*, to be *computational replicas* they must be *recomputations* and *reproductions* and to be *total replicas* they must be *repetitions* and *reproductions*.

Tables 4.5, 4.6 and 4.7 summarise the information we store to ensure scientific, computational and total replication respectively. The details included are straightforward unions of reproduction and rerunning, recomputing and repeating, respectively.

Table 4.5: Information stored to assert workflow scientific-replication for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type InPorts OutPorts	-	Type Name	-	-	Status
Data	“	“	Type Storage-Name	“	“	Status Data-Summary
Control (misc)	“	“	Type	“	“	-

4.5 Workflow Verification

We now describe a workflow’s entire journey, from a logical graph template down to executed drops. The transformations from logical graph template to logical graph and physical graph stencil to physical graph are straightforward operations. Unrolling a logical graph to a physical graph template and partitioning a physical graph template into a physical graph stencil are significantly more complicated operations. Deploying and executing a physical graph stencil as a physical graph requires total communication

Table 4.6: Information stored to assert workflow computational-replication for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type InPorts OutPorts	Num-CPUs Fields	Type Rank Name	Node Island	Node-IP Island-IP	Status Trace
Data	“	Data-Volume FileNames Fields	Type Rank Storage-Name	“	“	Status Data-Summary
Control (misc)	“	Fields	Type Rank	“	“	-

Table 4.7: Information stored to assert workflow total-replication for different components.

Component-type	LGT	LG	PGS	PGT	PG	RG
Application	Type InPorts OutPorts	Num-CPUs Fields	Type Rank Name	-	-	Status
Data	“	Data-Volume Fields	Type Storage-Name	“	“	Status Data-Summary
Control (misc)	“	Fields	Type Rank	“	“	-

between drops growing $\mathcal{O}(E)$ in time. However, drops communicate in a fully distributed manner, maximising exploited parallelism. The final runtime BlockDAG construction takes $\mathcal{O}(V(D \log D) + E)$ time as explained in section 4.3. Table 4.8 summarises the runtime of each operation. Since processing

Table 4.8: A summary of asymptotic complexity of each workflow operation and BlockDAG construction.

Operation	Time	Communication	BlockDAG	Cumulative Total
LGT - LG	$\mathcal{O}(V + E)$	-	$\mathcal{O}(V(D \log D) + E)$	$\mathcal{O}(V + V(D \log D) + E)$
LG - PGT	$\mathcal{O}(V + E)$	-	$\mathcal{O}(V(D \log D) + E)$	$\mathcal{O}(2(V + V(D \log D) + E))$
PGT - PGS	$\mathcal{O}(V + E)$	-	$\mathcal{O}(V(D \log D) + E)$	$\mathcal{O}(3(V + V(D \log D) + E))$
PGS - PG	$\mathcal{O}(V + E)$	-	$\mathcal{O}(V(D \log D) + E)$	$\mathcal{O}(4(V + V(D \log D) + E))$
PG - RG	$\mathcal{O}(V + E)$	$\mathcal{O}(E)$	$\mathcal{O}(V(D \log D) + E)$	$\mathcal{O}(5(V + V(D \log D) + E) + E)$
Amortised Total	$\mathcal{O}(V + E)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(V + E)$

and ultimately executing a workflow is an $\mathcal{O}(V + E)$ operation, passing over the graph to build a BlockDAG imposes an additional $\mathcal{O}(D \log D)$ overhead, the cost of hashing each workflow component, which amortises to a constant-time overhead for a single graph against the cost of processing and running the workflow in the first place. Building the BlockDAG at each layer does not increase each operation’s asymptotic complexity, and the total amortised constant-time complexity of imposing any reproducibility check is a requirement for ska-scale deployment. Testing for tenet compliance with a previous workflow execution at any abstraction level then takes $\Theta(1)$ time; the complexity of comparing two signatures.

4.6 Conclusion

This chapter explains our approach to implementing a verification framework for our previously defined reproducibility tenets (Chapter 3). We achieve this feat by enacting a component mapping from DALiuGE components to equivalents in our workflow model (Chapter 2), then embedding this information into a BlockDAG, a blockchain-inspired data structure at each stage in a DALiuGE workflow’s lifecycle. The result is an automatic and amortised constant-time method to verify if two workflow executions are reruns, repetitions, recomputations, reproduction or replicas. We now demonstrate our approach’s practicality on an example workflow, testing our DALiuGE implementation with a native implementation of our BlockDAG technique.

Chapter 5

Example Numerical Workflow

This chapter presents the validation of our scientific reproducibility verification method. We do so with native and DALiuGE implementations of a low-pass filter via fast Fourier transform and direct convolution. Section 5.1 introduces the example workflows. A low-pass filter is an ideal candidate workflow for several reasons. Signal-filtering is simple to describe and understand, data-driven, several well-known computational methods exist to compute it and is numerically sensitive. Therefore, a low-pass filter is an ample sample application to verify our reproducibility tenets’ efficacy, both in principle and practice.

Sections 5.2 to 5.6 present several experiments performed on our workflow to demonstrate direct application of our reproducibility tenets as defined on our scale-agnostic data-processing workflow model in Chapter 3. We chiefly aim to establish our approach’s capability to discern successful and unsuccessful rerun, repeated, reproduced and replicated workflows between native workflows, DALiuGE workflows and across these two different platforms. We leverage our reproducibility tenets’ well-structured nature in structuring our experimental approach, with each tenet having an individual methodology and a results discussion.

5.1 Workflow Description

We introduce our example workflow. A low-pass filter takes an assumedly noisy signal with components in many frequencies, convolves the data points with a window series resulting in a clean signal with no components from lower or high-frequency bands than required. Figure 5.2 provides plots of example data-files. We direct an unfamiliar reader to Smith’s Introduction to Digital Filters [192] as an excellent introductory source and basis for our workflow implementation. One can naïvely convolve a signal in $\mathcal{O}(N^2)$ time by multiplying each signal data-point with each window data-point, utilising a fast Fourier transform (FFT) permits an $\mathcal{O}(N \log N)$ approach [193]. Our workflows generate signal and window data, performs a low-pass filter operation and writes the result to a file. The input signal is real-valued, a power of two in length (for simplicity when using FFT methods) and composed of several sine-waves of varying frequency. The filter window is a Hann window, also a power of two in length. Figure 5.1 provides a UML depiction of our workflow comprising a single logical task and three data-artefacts for our workflow model described in Chapter 2.

We implement this workflow using three FFT libraries and a single direct-convolution approach

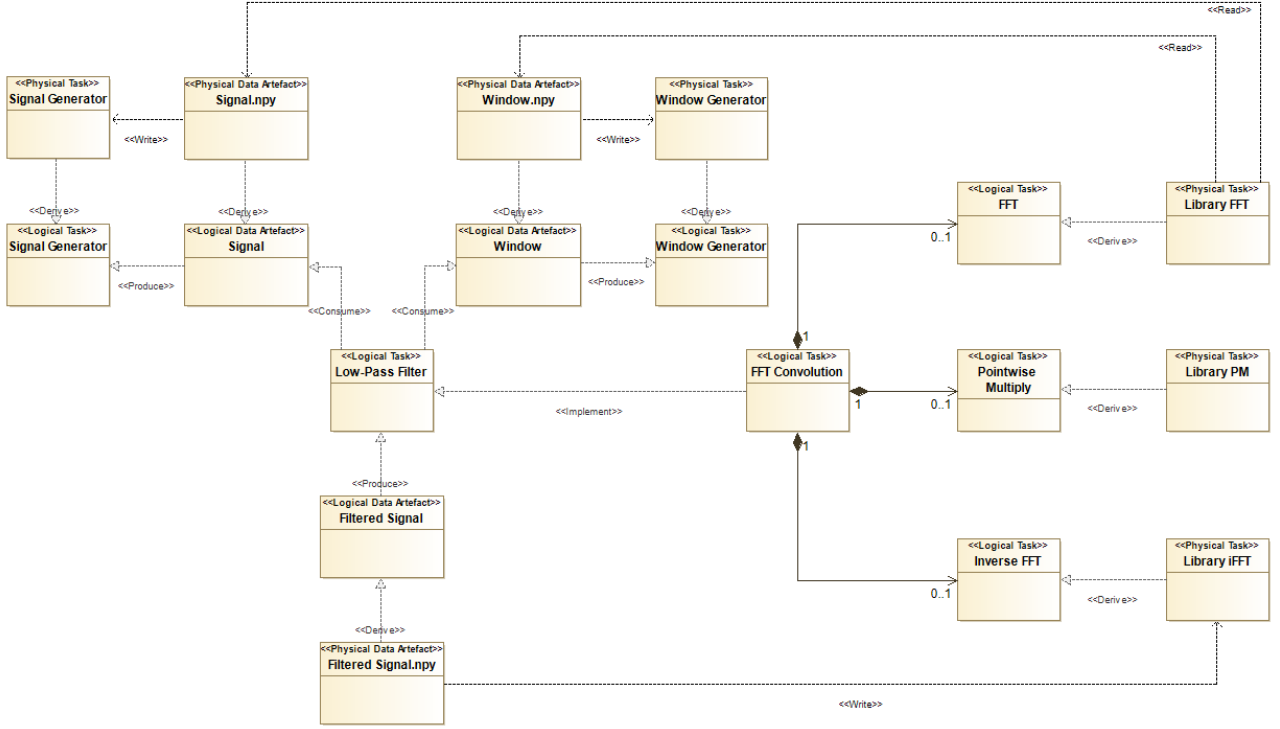


Figure 5.1: UML description of our low-pass filter via FFT convolution workflow. Each library specific implementation swaps out the three physical tasks.

based on the following libraries:

- Numpy FFT - Numpy is a de-facto standard high-performance numerical library written in C with Python bindings [194]
- PyFFTW - Based on the popular Fastest Fourier Transform in the West library [195], [196]
- ScikitCuda / PyCuda - A GPU-accelerated implementation of linear algebra routines [197]–[199]
- Numpy Convolution - A time-domain based point-wise implementation. [194]

Our example code on Github¹ implements this workflow in Python directly. Each library effectively replaces the physical tasks in our workflow, and the NumPy direct-convolution changes the logical task of FFT convolution to point-wise convolution.

In DALiuGE, we specify identical workflows using the Editor for the Astronomical Graph Language Environment (EAGLE)² using custom python components³. Figure 5.3 presents an example DALiuGE workflow using the PyCuda library where the comparison between our UML native workflows is startlingly similar. Concerning our workflow mapping, the DALiuGE workflow is a logical workflow; DALiuGE translates the logical workflow to a physical workflow. Changing the properties of individual nodes changes the eventual physical components instanced and the simplistic nature of this sequential test means unrolling does not affect workflow structure. All logical tasks are Python components,

¹<https://github.com/pritchardn/simplelowpass>

²<https://github.com/ICRAR/EAGLE>

³https://github.com/pritchardn/daliuge/blob/master/daliuge-common/dlg/common/reproducibility/apps_lowpass.py

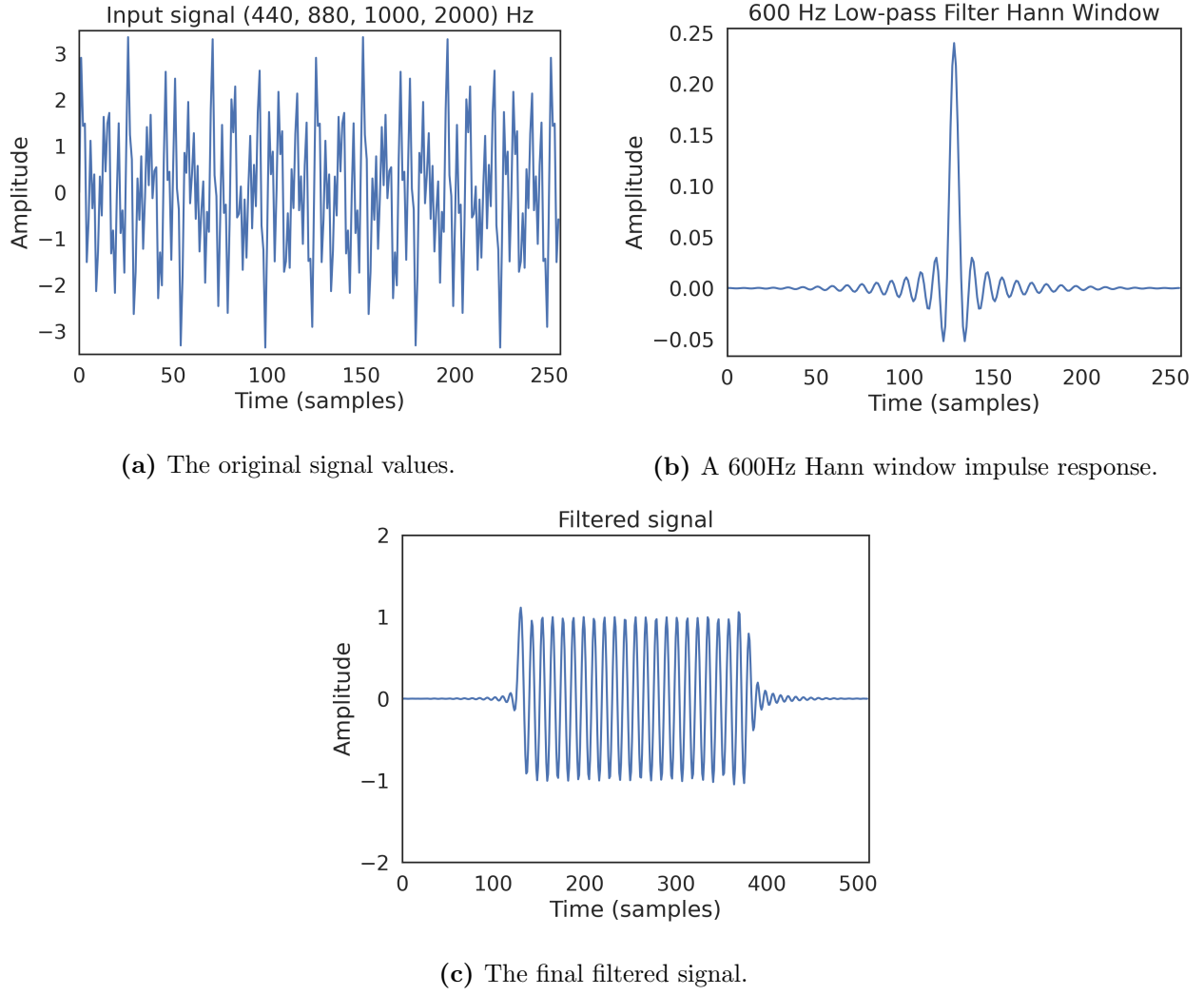


Figure 5.2: Representation of the entire low-pass filter process (Note the increase in time-series length in the final signal).

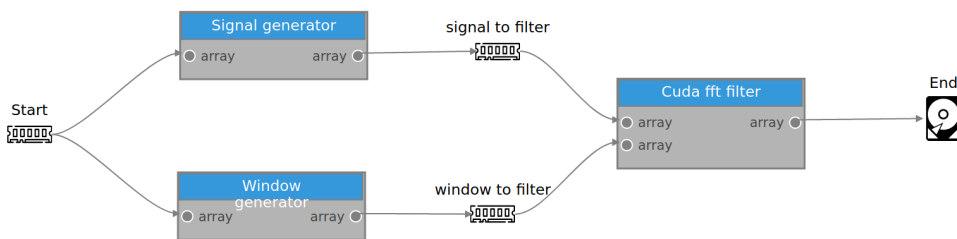


Figure 5.3: DALiuGE workflow for a CUDA based low-pass filter via FFT convolution created in the Editor for the Astronomical Graph Language Environment (EAGLE)

Testing workflow reruns is straightforward with our BlockDAG approach, and we expect all workflows to result in the same workflow signature, regardless of implementation or workflow management system.

5.2.1 Method

We unroll and execute each DALiuGE and native workflow on the same standard machine on two different data sets. The BlockDAG construction methods specified in Chapter 4 handle collecting runtime information from each component and assembling them to a single signature. Workflow executions with the same signature are reruns and are not otherwise.

5.2.2 Results

We present the hash values for each workflow execution in table 5.1. Clearly, across both execution platforms, all workflows match independent of the data set ^{4,5}. Notice that even the physically different

Table 5.1: A comparison of workflow implementations as reruns. All but the NumPy Point-wise workflow are reruns. D1 and D2 indicate the processed dataset.

Library / Workflow Trial	DALiuGE (D1)	Native (D1)	DALiuGE (D2)	Native (D2)
NumPy FFT	dfc34	dfc34	dfc34	dfc34
PyFFTW	dfc34	dfc34	dfc34	dfc34
PyCuda	dfc34	dfc34	dfc34	dfc34
NumPy Point-wise	dfc34	dfc34	dfc34	dfc34

Cuda implementations are reruns. Depending on interpretation, there is a good argument to be made for the point-wise workflows not to be considered reruns. This change requires including finer-grained information into the workflow provenance data. In summary, rerunning workflows allows logical workflow reuse independent of the underlying computing resource or implementation.

5.3 Repeat

We task our low-pass filter implementations with filtering several noisy input signals to demonstrate the value of repeat workflow executions. This experiment tests each filtering method’s effectiveness by repeat trials and demonstrates our ability to demarcate physical workflow changes. We expect each workflow to result in a different signature, differing between DALiuGE and native executions. However, signatures for each repeated trial should match despite changing input data.

5.3.1 Method

Data preparation We generate our input signal as a combination of several sine signals and add noise by adding a uniformly randomly varying amplitude component to the 1200Hz frequency band. We seed our random number generator with the trial number to ensure each trial signal is different but reproducible and also process the same signal without noise as a control.

⁴https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/rerun.csv

⁵https://github.com/pritchardn/simplelowpass/blob/merkleTrees/DALiuGE_results/rerun.csv

Effectiveness metric We compute the normalised cross-correlation (NCC) [200] divided by the signal length between the control and each noisy trial for each method to compare our methods. The resulting metric gives the probability that any data point in a filtered signal matches the corresponding ground-truth value. This metric is relative, and hence the length of our final signal data series will determine the numerical precision we need to make an accurate judgement ($1 / \text{signal-length}$). In all trials, the final signal length is 512 data points, and hence three decimal points of accuracy are sufficient. We can perform such a direct comparison (instead of more sophisticated spectral methods) since we can guarantee that the input signal does not shift in time for any trial and any method tested. We then average the NCC over all trials to arrive at a precise metric conveying the proportion of data points that matched the ground truth filtered signal over all trials completed.

5.3.2 Results

Table 5.2 presents our results for ten random trials for all workflow methods ⁶. In theory, a perfect filter would have an NCC of precisely one. Table 5.3 presents a truncated table of workflow signatures for each native ⁷ and DALiuGE ⁸ trial. All but the Cuda based workflows achieve the same performance level, and none are a perfect filter. Moreover, all trials of identical physical workflows match in signature.

Table 5.2: Averaged Normalised Cross Correlation (NCC) values for each filter implementation.

Method	Normalised Cross Correlation (NCC)
Ground Truth	1.000
PyCuda	0.673
PyFFTW	0.890
Numpy FFT	0.890
Numpy Point-wise	0.890

⁶https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/repeat1.csv

⁷https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/repeat.csv

⁸https://github.com/pritchardn/simplelowpass/blob/merkleTrees/DALiuGE_results/repeat.csv

Table 5.3: Workflow signatures for native and DALiuGE repeated trial executions. We truncate hash values for brevity.

Workflow / Trial	1	2	3	4	5	6	7	8	9	10
PyCuda (Native)	65f0d	65f0d	65f0d	65f0d	65f0d	65f0d	65f0d	65f0d	65f0d	65f0d
PyFFTW (Native)	ea4db	ea4db	ea4db	ea4db	ea4db	ea4db	ea4db	ea4db	ea4db	ea4db
NumPy FFT (Native)	d8465	d8465	d8465	d8465	d8465	d8465	d8465	d8465	d8465	d8465
NumPy Point-wise (Native)	26844	26844	26844	26844	26844	26844	26844	26844	26844	26844
PyCuda (DALiuGE)	2a524	2a524	2a524	2a524	2a524	2a524	2a524	2a524	2a524	2a524
PyFFTW (DALiuGE)	7ae30	7ae30	7ae30	7ae30	7ae30	7ae30	7ae30	7ae30	7ae30	7ae30
NumPy FFT (DALiuGE)	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7	5a3a7
NumPy Point-wise (DALiuGE)	b1570	b1570	b1570	b1570	b1570	b1570	b1570	b1570	b1570	b1570

By certifying repeated trials as formal repetitions for eight workflow implementations, we concretely establish our definition of repetition and subsequent implementations and discover a shortcoming in the Cuda based implementation. Perhaps investigating the reproducibility of each workflow may reveal some hidden reasoning.

5.4 Recompute

Recomputation embodies the most careful method reproducibility approach, where two workflow executions must match on all physical tasks exactly. Here, we test for recomputation on and between two hardware platforms to demonstrate the ability to demarcate subtle hardware differences.

Method We run all of our low-pass filter workflows on two machines, with the same configuration parameters used in our rerun test. We present data comparing the results from an Ubuntu-based machine equipped with an Nvidia GTX 1070 and an Intel i7-6700 CPU ⁹ and a Windows-based machine equipped with two Nvidia GTX 980s and an Intel i7-5900 CPU ¹⁰. Each drop adds the function arguments and hardware information to their signatures. If two computation signatures match, we are confident in the execution of precisely the same code on precisely the same hardware. DALiuGE does not support Windows natively, and hence we only attempt to recompute our native workflows in Windows.

Results Tables 5.4 and 5.5 present our workflow signatures for five trials. There was no successful attempt at recomputing any workflow. Attempts to recompute data-processing workflows help build trust in workflow execution systems, an engineering task, but is perhaps less helpful for investigating the integrity of the underlying tested science.

Table 5.4: Workflow signatures for Linux machine recomputations. We truncate hash values for brevity.

Workflow (Linux) / Trial	1	2	3	4
PyCuda (Native)	ed5c5	de324	77dce	6c80e
PyFFTW (Native)	afba6	91e85	61004	3e83a
NumPy FFT (Native)	46160	02c41	b7627	2e3bf
NumPy Point-wise (Native)	7bc29	97762	a23c2	05922
PyCuda (DALiuGE)	a7f8c	dad7d	74258	25ea0
PyFFTW (DALiuGE)	979ed	d4be1	63939	529eb
NumPy FFT (DALiuGE)	7bb10	aaf70	a7e35	56c9e
NumPy Point-wise (DALiuGE)	ed377	f5731	03413	ee271

5.5 Reproduce

Our reproducibility tenet focuses solely on data-artefacts. Motivated to uncover a discrepancy between our workflow implementations, we task our workflows with filtering the same signals. This rather

⁹https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/system.json

¹⁰https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/windows/system.json

Table 5.5: Workflow signatures for Windows machine recomputations. We truncate hash values for brevity.

Workflow (Windows) / Trial	1	2	3	4
PyCuda (Native)	1628a	2daa3	04231	b0002
PyFFTW (Native)	df2b6	4ea2e	b16fc	508d9
NumPy FFT (Native)	985e8	e56e9	0f51b	35412
NumPy Point-wise (Native)	ba031	fb3af	3a471	1696b

basic experiment demonstrates the task of iterating upon workflow components; a library that may be more performant should ideally not impact numerical outputs. A reproduction test will determine if a change in logical task implementation impacts known results.

Method We create several input signals with varying higher frequency components, signal lengths and window lengths. We accumulate and average the normalised cross-correlation with every other method for each workflow and each input signal trial. The normalised cross-correlation provides a machine-precise metric for how similar each method’s results are across several different input signals. For a final signal comprised of 512 data points, three decimal places are sufficient to establish a perfect reproduction, as is the case in all of our trials.

Results Table 5.6 contains the averaged normalised cross-correlation over all trials between all methods. We see that all but Cuda-based methods reproduce the same results according to our expectation that these methods are equivalent. Moreover, there is no difference between running a workflow through DALiuGE or natively. Table 5.7 contains workflow signatures for all executed trials.

Table 5.6: Normalised cross-correlation between all filter implementations.

Method (NCC)	PyCuda	PyFFTW	Numpy FFT	Numpy FFT
PyCuda	1.000			
PyFFTW	0.707	1.000		
Numpy FFT	0.707	1.000	1.000	
Numpy Point-wise	0.707	1.000	1.000	1.000

Results for native ¹¹ and DALiuGE ¹² trial executions are available online. While each workflow results in a different signature and all methods produce different signatures, we reproduce our results between native and DALiuGE workflows for all but the NumPy point-wise method. In summary, specifically targeting data reproducibility allows our framework to be compatible with prior works considering reproducibility across a wide variety of fields [1], accepts current practical reproducibility methods from Software Engineering [45].

5.6 Replicate

We demonstrate the differences between scientific, computational and total workflow replication. Our replication tenets are conveniently combinations of previously discussed tenets. Summarised again here,

¹¹https://github.com/pritchardn/simplelowpass/blob/merkleTrees/final_results/reproduce.csv

¹²https://github.com/pritchardn/simplelowpass/blob/merkleTrees/DALiuGE_results/reproduce.csv

Table 5.7: Workflow signatures for native and DALiuGE reproduction trial executions. We truncate hash values for brevity.

Workflow / Trial	1	2	3	4
PyCuda (Native)	b008b	e8af1	7e430	4c051
PyFFTW (Native)	38c04	97438	8311f	09fbd
NumPy FFT (Native)	0255f	3456a	eb1cd	1c8aa
NumPy Point-wise (Native)	21b56	f3f7e	83c91	81d63
PyCuda (DALiuGE)	b008b	e8af1	7e430	4c051
PyFFTW (DALiuGE)	38c04	97438	8311f	09fbd
NumPy FFT (DALiuGE)	0255f	3456a	eb1cd	1c8aa
NumPy Point-wise (DALiuGE)	9f322	27dc1	eb4ba	09ff4

for simplicity, a scientific replication requires rerunning and reproducing a workflow, a computational replication requires recomputing and reproducing a workflow and a total replication requires repeating and reproducing a workflow. We demonstrate our definitions by attempting to replicate our previous experiments on different hardware.

Method We provide Python scripts to produce all digital artefacts used throughout this thesis, including running our repetition and reproduction tests and then compares these outputs with the values presented in this thesis¹³. We deploy DALiuGE workflows manually, however. To discriminate between a scientific, computational, and total replication, we run all workflow examples natively on our previously specified Linux and Windows machines and with DALiuGE on our Linux machine. Testing for a computational replication includes all available details, a total replication includes specific software details, and a scientific replication includes only coarse software information. If the result files and environment signatures match, the replication attempt is successful and fails otherwise. We use the same two machines specified in Section 5.4.

Results Table 5.8 summarises all scientific replication execution signatures for all workflows and Tables 5.9 and 5.10 contain equivalent information for computational and total replication attempts respectively. We successfully scientifically replicate all trials for all but the point-wise workflows between native Linux and DALiuGE executions. Otherwise, all other trials for all methods on all platforms are not computational or total replicas.

¹³<https://github.com/pritchardn/simplelowpass/blob/merkleTrees/src/replicate.py>

Table 5.8: Workflow scientific replication attempts for all workflows and all machines.

Workflow / Trial	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PyCuda (Linux)	54e8a	2aab	b02ac	fb569	3fd1c	ec5bc	e97ed	804d6	9381a	1f267	bdd12	0a85b	03991	fa337
PyFFTW (Linux)	2ae29	d5454	de606	0fa7	5ee54	053a6	fbca0	f0432	62bb9	79d34	642f7	f83f5	5afbc	69969
NumPy FFT (Linux)	784bd	0d4cb	73326	67fda	42b2d	5fbec	2ea57	95aab	4dea	48bd9	763b4	15cba	f4146	67115
NumPy Point-wise (Linux)	f43fa	170d6	6acf6	81580	e094a	20828	4f035	b27ab	b583f	19524	d6414	cac7a	10490	13330
PyCuda (Windows)	65ebf	edc9c	00b2e	6781a	e4ad1	55f09	0d663	c4c64	bb8c3	b49cf	174aa	e3b8a	f9414	b5a17
PyFFTW (Windows)	0c112	0cb0a	372c3	302c0	42cbe	45786	3af66	a1b5d	1ff74	9c5a4	94d27	3f10c	09f4a	aebe9
NumPy FFT (Windows)	eb76f	9bb1e	600c8	c3c44	423b3	1fdf5	b0888	c0b02	487bb	9b03d	13888	018fd	90e1f	276e3
NumPy Point-wise (Windows)	18c78	9fc25	f82fd	940aa	dd6d3	4beb0	2af99	405f5	56ebb	36316	b9a74	11218	c268d	93711
PyCuda (DALiuGE)	54e8a	2aab	b02ac	fb569	b6914	8bbe6	22389	43a22	1608c	bb414	81b9f	5aa27	b23ac	e4785
PyFFTW (DALiuGE)	2ae29	d5454	de606	0fa7	775ca	a33b5	45d0a	1b01e	a19bc	5b62d	d5b6e	94833	ad6fa	ace56
NumPy FFT (DALiuGE)	784bd	0d4cb	73326	67fda	b6b3e	6a2df	17917	496fa	92f24	5bbe5	5bd97	a0929	5f7a0	b058c
NumPy Point-wise (DALiuGE)	ff7a0	1a078	6dd02	2bd2a	ff7a0	de74e	5e164	9261a	f7a4c	0b992	97106	fed3b	9bddb	3a338

Table 5.9: Workflow computational replication attempts for all workflows and all machines.

Workflow / Trial	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PyCuda (Linux)	3c671	db3e7	f349b	5fa94	b87e0	8e9a0	872fc	3f627	0aabd	9ded5	facff	009cb	8310d	9a23d
PyFFTW (Linux)	6e4a6	47617	dfb06	11773	b9690	5af2a	dc54b	b7fd3	a1c69	a0bba	5691b	343c3	afd19	6d052
NumPy FFT (Linux)	8dda3	efc50	e5a91	09ab9	205c3	ac50d	4f68a	8a4eb	cc9b5	17c97	7008b	188b2	b3f66	24c7a
NumPy Point-wise (Linux)	83846	8a004	35045	80b02	1a0b3	815be	2b0cf	46d5a	20495	f856b	d13bc	82c4a	23855	4241b
PyCuda (Windows)	010ad	03775	2ee17	32770	b40cb	fe6fa	bed63	bbeea	acc3f	1dae3	f254c	b1d56	b38e9	30d72
PyFFTW (Windows)	44adf	bb9c1	08efd	08479	c9517	02521	e062c	91dad	0a8e3	a24ee	207ab	5db57	2308e	9983a
NumPy FFT (Windows)	93d3b	f9ee3	4fa26	30ce7	72f4c	55108	563a2	3cb18	2a097	6e98c	e1f21	0a9dc	e2395	94173
NumPy Point-wise (Windows)	9a40e	cfb54	8c978	c15cb	5a786	d344b	3ee77	cc5bd	e424b	9207d	3c182	0c62e	fa93a	7e5c0
PyCuda (DALiuGE)	c0fle	03909	f5e20	6e4a3	17210	dafc5	0d735	60005	9fb61	aaafe	d2329	3c604	bc363	5fe58
PyFFTW (DALiuGE)	18d3c	703b5	9b1d7	d576d	19268	78370	11c77	63341	602fe	effbe	c27d3	b4691	4917c	cebdd
NumPy FFT (DALiuGE)	95af1	a67ac	7b9b0	4959b	83eb2	6f956	45889	56f09	bf946	bdb22	5ed91	16464	872dc	176c1
NumPy Point-wise (DALiuGE)	0cdde	beb94	27013	3f19e	0cdde	09572	f06b2	8ad26	db316	3852d	e44d9	8a3dd	1a6af	dc0bb

Table 5.10: Workflow total replication attempts for all workflows and all machines.

Workflow / Trial	1	2	3	4	5	6	7	8	9	10	11	12	13	14
PyCuda (Linux)	2a6da	9ef65	f531f	ad816	869b3	da299	f6f4c	f4aec	2d4c9	025cc	58881	04231	44874	d4d73
PyFFTW (Linux)	0737f	966f5	71f2e	00d3b	97c9d	c4667	a23fb	48778	2e5d6	b00e6	3bd53	0e382	dbbf6	c0091
NumPy FFT (Linux)	83fd8	8689b	bc1d4	e128f	0bd0b	7c114	69f9e	80fd4	1e06e	452bf	529e8	40150	f2114	6ed17
NumPy Point-wise (Linux)	912f7	c3788	95ed4	8328c	cb404	ed8be	4f6b6	86e31	e025a	a3393	15882	316be	71959	c0756
PyCuda (Windows)	16c15	34d02	647a3	330ed	e7ad4	4709b	fb7b5	6c7c4	65123	20c39	98eca	2d754	bef0a	dc87a
PyFFTW (Windows)	05476	b8f72	8fc48	6121f	06b14	e2830	5dea8	a5de2	35025	6ad38	0f25e	3fd8f	d7d17	7f57e
NumPy FFT (Windows)	6ca24	11136	38322	ff36e	de1f0	d58a4	a651d	f3dba	55db1	dbf54	2ef3	01e8f	d602c	cff32
NumPy Point-wise (Windows)	c0ac9	de7ca	8b9c3	b7e21	b8a6e	26f0d	b76a9	e9d80	bf10e	0520a	73001	6c95e	d6d77	4a878
PyCuda (DALiuGE)	9b6b0	d832f	045b0	a82bf	24a49	7af8c	08ed0	9e10e	9f951	5397d	373d6	945e8	b8c32	9b7ab
PyFFTW (DALiuGE)	1bacf	a5509	c4a8c	14e79	9525e	b13d0	6c611	96047	86018	f7d77	117eb	35550	3a128	3cd4d
NumPy FFT (DALiuGE)	5b532	2ebff	6c80f	c2ce9	1eace	e0a8a	47f5f	43ed1	09d11	c39ae	4a291	8ee06	40b8d	deb4b
NumPy Point-wise (DALiuGE)	9e7ee	c1fc0	22e6c	948f9	9e7ee	29042	99569	36d9a	094b9	46d59	6ae0b	63b7f	0da2c	5b8a8

If we consider our repetition test, however, where the actual final scientific output is the averaged normalised cross-correlation of our workflow executions, we find that we would scientifically replicate the results of this experiment on our Windows platform. Doing so, however, requires embedding this post-processing step into our workflow; the decision of where to demarcate a workflow's end is now testable and meaningful. We have accurately characterised the difference between various implementations of a low-pass filter in a reliable, machine-precise manner.

Establishing apparent differences between workflow task and data-artefact replication allows us to build confidence in the logic, execution or implementation precisely when testing for a scientific, computational or total replication, respectively.

5.7 Discussion

Through a low-pass filter example workflow, we make our definitions concrete. Table 5.11 summarises our results. We successfully rerun all methods on all platforms, reproduce all results (apart from point-wise methods) on Linux and DALiuGE platforms, and all other workflows are unique. Rerunning tests

Table 5.11: Attempt to replicate experimental results in the original environment and on a different machine.

Method	Rerun	Repeat	Recompute	Reproduce ¹⁴	Scientific Replication	Computational Replication	Total Replication
Linux	✓	✗	✗	✓	✓	✗	✓
Windows	✓	✗	✗	✗	✗	✗	✗
DALiuGE	✓	✗	✗	✓	✓	✗	✗

the execution of a logical workflow, and in our example, we distinguish between workflows with identical dependencies and find similarity between workflows with different hardware requirements. Repetition tests if we can bundle workflow execution results, which in our investigation of multiple low-pass filter implementations, we find our GPU based workflow to be the least accurate. Recomputation catches machine-specific oddities and details. Our investigation found it impossible to recompute our low-pass filter workflow precisely, even on identical hardware, demonstrating the difference between repetition and recomputation. Reproduction tests if multiple workflow designs and implementations produce identical results. Our GPU workflow example produces different results from all other methods. We define replication as three combinations of previous tenets. Scientific replication tests if multiple implementations of the same logical workflow generate identical results; our example achieves comparison across operating system, hardware differences and workflow management system. Computational replication tests the outputs of a single implementation over multiple hardware configurations; our example demonstrates that even slight differences can yield different results. Finally, total replication allows a result comparison of any equivalent logical workflow implementation, which, in our example, establishes differences with the GPU model.

We have not yet discussed the combination of and deliberate failing of reproducibility tenets. For instance, a failed rerun and successful reproduction imply that a logically different workflow achieves an equivalent overall computation. Although it is beyond the scope of this example, modular

¹⁴not point-wise method

workflows allow modular reproducibility considerations, which, when tested aptly, may promote reuse of well-known sub-workflow designs and implementations [113]. Furthermore, one may reasonably ask, how is the testing of workflow standards any different from building computational workflows to a particular technical standard?

The critical observation is that we can test to refute any of these tenets at will. Successful reproducibility tests may build confidence in our workflow implementation and tooling, but ideally, a substantial scientific claim holds up over *any* adequate implementation. For a valid scientific verification, we propose the failure to rerun and recompute, but success in reproducing only the output of a given workflow is a universal computation. By shifting focus to testing rather than tooling, we can use our tenets to build confidence in a particular implementation and computation only to strip them away to aid in making a scientific claim that holds over any implementation. Constructing a ‘backwards-facing’, testing driven view of computational workflow reproducibility aligns sound software practices with the philosophical ideas driving science at its core, permitting better integration of computational data-processing into scientific processes.

5.8 Conclusion

This chapter enacted our test-driven approach to computational workflow reproducibility to define seven reproducibility tenets. We demonstrate each tenet on several DALiuGE and native workflows, demonstrating the flexibility of our workflow model. A test-driven approach to computational workflow reproducibility powered by fundamental blockchain technologies goes some way to addressing scientific reproducibility concerns for data-processing sciences.

Chapter 6

Conclusions and Outlook

This chapter summarises our principal contributions in their presented order, discussing findings and providing direction for future work where appropriate. We omit Chapter 1 as it only introduces pre-requisite background and outline material.

Chapter 2 Scale-Agnostic Scientific Data Processing

This chapter introduces a scale-agnostic arbitrary workflow model. This model ensures that further reproducibility reasoning applied to the model applies to any mapped workflow management system. We subsequently map seven well-known workflow management systems to our model to demonstrate its flexibility and frame our reproducibility discussion in a broader context than a single toolset. Service-based workflow management systems propose a particularly challenging mapping, and not all systems model data artefacts in an abstract sense. Moreover, the UML description permits the native implementation of specific workflows, a unique property. Separating logical and physical workflow components introduces expressiveness into the model and accurately captures the workflow life-cycle from conception to execution. Employing a workflow model makes subsequent reproducibility reasoning system agnostic. Enacting and extending our present mappings in addition to testing our model against contemporary alternatives is a clear future direction.

Contributions

1. A novel scale-agnostic data processing workflow model.
2. Maps seven well known computational workflow management systems to our abstract model, providing a baseline model for commensurate discussion.

Future Work

- Mapping more workflow management systems to this model.
- Extending the specificity of component descriptions to include more details.

Chapter 3 Reproducible Scientific Data Processing

This chapter formally defines seven well-discussed reproducibility tenets for our scale-agnostic data-processing workflow model as a collection of testable assertions. Historical developments from the philosophy of science and contemporary literature guide our approach to frame these tenets as testable assertions on past workflow executions, complementing contemporary approaches to improve workflow reproducibility. Defining reproducibility tenets on our previously introduced workflow model makes our definitions actionable and widely applicable to any mapped workflow management system. For each tenet, we provide a clear definition, justification and a few motivating examples. Our definitions are broadly compatible with currently accepted definitions. However, we expand replication as a combination of prior tenets providing internal consistency within a systematic framework. There may be room for more nuanced tenet definitions and broader application to non-computational workflows.

Contributions

1. Definition of *rerunning* computational workflows as a loose constraint on methodology.
2. Definition of *repeating* computational workflows as a strong constraint on methodology.
3. Definition of *recomputing* computational workflows as a maximally precise constraint on methodology.
4. Definition of *reproducing* computational workflows as a constraint on scientific information.
5. Definition of *replicating* computational workflows as a variable combination of method and data constraints. A *scientific replica* is a combination of rerunning and reproduction, a *computational replica* is a combination of recomputing and reproduction, and a *total replica* is a combination of repetition and reproduction.

Future Work

- Can we include ontological reasoning into our reproducibility model and would this be beneficial?
- Do we need separate consideration for stochastic phenomena? Effectively testing *partial* compliance with a particular tenet.

Chapter 4 Methods

This chapter presents a blockchain-inspired approach to enacting our reproducibility tenets. This approach builds a BlockDAG from workflow component descriptions and runtime information. Implementation in the DALiuGE workflow management system touches every stage in a DALiuGE workflow life-cycle (or abstraction level), including centralised and decentralised phases. We also summarise what information each component type includes in its block for each reproducibility tenet and at each abstraction level. Our approach provides a novel constant-time scientific integrity verification mechanism that incurs an amortised constant-time impact on workflow enactment, a quality required for deployment at SKA scale.

Contributions

1. A novel blockchain-inspired data structure, the BlockDAG.
2. An amortised constant time scientific verification for computation workflow data.

Future Work

- Extending our component information requirements to include pre-existing formal provenance ontologies.
- Conducting a more extensive technical review of existing workflow management systems and their reproducibility enabling capabilities.

Chapter 5 Example Numerical Workflow

This chapter evaluates our workflow verification mechanism with several low-pass filter workflows as an example application. Our implementations utilise a variety of baseline libraries (PyCuda, FFTW and NumPy). We also compare our DALiuGE workflow with native implementations. We demonstrate the ability to assert workflow reruns across workflows utilising different physical components and across native and managed workflow implementations. Then, we demonstrate the value of repeating workflow executions to increase statistical power. Uncover the inability for Cuda based GPU implementations to reproduce the results of their CPU counter-parts and show the difficulty, but not impossibility, in replicating workflow behaviour across different hardware environments.

To our knowledge, this is the first formal example of workflow reproduction in both a workflow management system and native implementation and demonstrates the efficacy of our workflow model, reproducibility tenets and blockchain-inspired workflow execution verification mechanism.

Contributions

1. Verification of our workflow model through native workflow implementation.
2. Verification of our workflow mapping to the DALiuGE workflow management system.
3. Practically demonstrate our reproducibility tenets where we uncover differences and equivalences between several numerical libraries.

Future Work

- One should investigate our approach's scalability in handling workflows across multiple machines.
- Investigating how our approach handles inherently stochastic workflows will provide a necessary robustness test for our tenet definitions.

Closing Remarks

The ‘Reproducibility Crisis’ rocking the wider scientific world, we argue, is not so much a crisis but a necessary and arguably inevitable development owing to science’s success. Vast increases in our ability to process information facilitated by broad access to computing improve our capacity for science and strains the systems constituting science. Integrating these new capabilities into scientific communities, then, is an un-intuitively expected challenge.

Addressing computational scientific reproducibility should be commensurate with broader academic interpretations. Current attempts to achieve computational workflow reproducibility in science focuses on assuring future reproducibility. These approaches are reminiscent of the ultimately failed attempts by Logical Positivists at inferring scientific rationality [201]. To this end, we created a computational data-processing workflow model, mapped seven well-known workflow management systems to it and propose a reproducibility framework aiming to safely refute attempts to rerun, repeat, recompute, reproduce or replicate prior works. We then implement our testing using a novel blockchain-inspired method in the DALiuGE workflow management system, verifying our approach’s efficacy by investigating the reproducibility of several low-pass filter workflows. This thesis reframes the quest to merge computational and scientific reproducibility from attempts to infer future workflow reproducibility from technical standards to applying tests on past workflow executions defined in a system-agnostic manner. Focussing on the tests and test-process allows scientific comparison of works independent of tooling, thus complementing current efforts to ensure reproducibility by providing the capacity to test them. We invite others to extend and manipulate our mappings and implementations to enable verification and comparison of workflows and their underlying implementations.

Computational workflow management is at the core of the Square Kilometre Array (SKA) development and demands performance and scientific integration at a level previously inconceivable and currently untenable. So many people’s efforts over so many years towards implementing the SKA justifies a novel answer to scientific reproducibility. This thesis embodies an approach to scientific reproducibility at SKA scale. However, we believe that our approach is more widely applicable with potential use in any data-intensive scientific endeavour. We boldly aim to contribute to the scientific process itself on behalf of one of the most extensive ongoing global science projects.

Bibliography

- [1] F. Fidler and J. Wilcox, “Reproducibility of scientific results,” in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Winter 2018, Metaphysics Research Lab, Stanford University, 2018. [Online]. Available: <https://plato.stanford.edu/archives/win2018/entries/scientific-reproducibility/>.
- [2] S. van de Sandt, A. Lavasa, S. Dallmeier-Tiessen, and V. Petras, “The definition of reuse,” *Data Sci. J.*, vol. 18, p. 22, 2019.
- [3] J. Ioannidis, “Why science is not necessarily self-correcting,” *Perspectives on Psychological Science*, vol. 7, pp. 645–654, 2012. DOI: [10.1177/1745691612464056](https://doi.org/10.1177/1745691612464056).
- [4] Open Science Collaboration, “Estimating the reproducibility of psychological science,” *Science*, vol. 349, no. 6251, aac4716–aac4716, Aug. 28, 2015, ISSN: 0036-8075, 1095-9203. DOI: [10.1126/science.aac4716](https://doi.org/10.1126/science.aac4716). [Online]. Available: <https://www.sciencemag.org/lookup/doi/10.1126/science.aac4716>.
- [5] C. F. Camerer, A. Dreber, E. Forsell, T.-H. Ho, J. Huber, M. Johannesson, M. Kirchler, J. Almenberg, A. Altmejd, T. Chan, E. Heikensten, F. Holzmeister, T. Imai, S. Isaksson, G. Nave, T. Pfeiffer, M. Razen, and H. Wu, “Evaluating replicability of laboratory experiments in economics,” *Science*, vol. 351, no. 6280, p. 5, Mar. 25, 2016.
- [6] Board of Governors of the Federal Reserve System, A. C. Chang, and P. Li, “Is economics research replicable? sixty published papers from thirteen journals say ”usually not”,” *Finance and Economics Discussion Series*, vol. 2015, no. 83, pp. 1–26, Oct. 2015, ISSN: 19362854. DOI: [10.17016/FEDS.2015.083](https://doi.org/10.17016/FEDS.2015.083). [Online]. Available: <http://www.federalreserve.gov/econresdata/feds/2015/files/2015083pap.pdf>.
- [7] R. D. Peng, “Reproducible research in computational science,” *Science*, vol. 334, no. 6060, pp. 1226–1227, Dec. 2, 2011, Publisher: American Association for the Advancement of Science Section: Perspective, ISSN: 0036-8075, 1095-9203. DOI: [10.1126/science.1213847](https://doi.org/10.1126/science.1213847). [Online]. Available: <http://science.sciencemag.org/content/334/6060/1226>.
- [8] R. Peng, “The reproducibility crisis in science: A statistical counterattack,” *Significance*, vol. 12, no. 3, pp. 30–32, 2015. DOI: <https://doi.org/10.1111/j.1740-9713.2015.00827.x>. [Online]. Available: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1740-9713.2015.00827.x>.

- [9] C. F. Camerer, A. Dreber, F. Holzmeister, T.-H. Ho, J. Huber, M. Johannesson, M. Kirchler, G. Nave, B. A. Nosek, T. Pfeiffer, A. Altmeld, N. Buttrick, T. Chan, Y. Chen, E. Forsell, A. Gampa, E. Heikensten, L. Hummer, T. Imai, S. Isaksson, D. Manfredi, J. Rose, E.-J. Wagenmakers, and H. Wu, “Evaluating the replicability of social science experiments in nature and science between 2010 and 2015,” *Nature Human Behaviour*, vol. 2, no. 9, pp. 637–644, Sep. 2018, ISSN: 2397-3374. DOI: [10.1038/s41562-018-0399-z](https://doi.org/10.1038/s41562-018-0399-z). [Online]. Available: <http://www.nature.com/articles/s41562-018-0399-z>.
- [10] C. G. Begley and J. P. A. Ioannidis, “Reproducibility in science,” *Circulation Research*, vol. 116, no. 1, pp. 116–126, 2015. DOI: [10.1161/CIRCRESAHA.114.303819](https://doi.org/10.1161/CIRCRESAHA.114.303819). [Online]. Available: <https://www.ahajournals.org/doi/abs/10.1161/CIRCRESAHA.114.303819>.
- [11] S. A. Iqbal, J. D. Wallach, M. J. Khoury, S. D. Schully, and J. P. A. Ioannidis, “Reproducible research practices and transparency across the biomedical literature,” *PLOS Biology*, vol. 14, no. 1, D. L. Vaux, Ed., e1002333, Jan. 4, 2016, ISSN: 1545-7885. DOI: [10.1371/journal.pbio.1002333](https://doi.org/10.1371/journal.pbio.1002333). [Online]. Available: <https://dx.plos.org/10.1371/journal.pbio.1002333>.
- [12] M. C. Makel and J. A. Plucker, “Facts are more important than novelty: Replication in the education sciences,” *Educational Researcher*, vol. 43, no. 6, pp. 304–316, Aug. 2014, ISSN: 0013-189X, 1935-102X. DOI: [10.3102/0013189X14545513](https://doi.org/10.3102/0013189X14545513). [Online]. Available: <http://journals.sagepub.com/doi/10.3102/0013189X14545513>.
- [13] M. Baker, “1,500 scientists lift the lid on reproducibility,” *Nature News*, vol. 533, no. 7604, p. 452, May 26, 2016. DOI: [10.1038/533452a](https://doi.org/10.1038/533452a). [Online]. Available: <http://www.nature.com/news/1-500-scientists-lift-the-lid-on-reproducibility-1.19970>.
- [14] S. N. Goodman, D. Fanelli, and J. P. A. Ioannidis, “What does research reproducibility mean?” *Science Translational Medicine*, vol. 8, no. 341, 341ps12–341ps12, 2016, Publisher: American Association for the Advancement of Science, ISSN: 1946-6234. DOI: [10.1126/scitranslmed.aaf5027](https://doi.org/10.1126/scitranslmed.aaf5027). [Online]. Available: <https://stm.sciencemag.org/content/8/341/341ps12>.
- [15] J. P. Mesirov, “Accessible reproducible research,” *Science*, vol. 327, no. 5964, pp. 415–416, 2010, Publisher: American Association for the Advancement of Science, ISSN: 0036-8075. DOI: [10.1126/science.1179653](https://doi.org/10.1126/science.1179653). [Online]. Available: <https://science.sciencemag.org/content/327/5964/415>.
- [16] J. F. Claerbout and M. Karrenbach, “Electronic documents give reproducible research a new meaning,” in *SEG Technical Program Expanded Abstracts 1992*, 2005, pp. 601–604. DOI: [10.1190/1.1822162](https://doi.org/10.1190/1.1822162). [Online]. Available: <https://library.seg.org/doi/abs/10.1190/1.1822162>.
- [17] R. Nuzzo, “How scientists fool themselves and how they can stop,” *Nature News*, vol. 526, no. 7572, p. 182, Oct. 8, 2015, Section: News Feature. DOI: [10.1038/526182a](https://doi.org/10.1038/526182a). [Online]. Available: <http://www.nature.com/news/how-scientists-fool-themselves-and-how-they-can-stop-1.18517>.
- [18] G. J. Lithgow, M. Driscoll, and P. Phillips, “A long journey to reproducible results,” *Nature*, vol. 548, no. 7668, pp. 387–388, Aug. 2017, ISSN: 0028-0836, 1476-4687. DOI: [10.1038/548387a](https://doi.org/10.1038/548387a). [Online]. Available: <http://www.nature.com/articles/548387a>.

- [19] R. M. Shiffrin, K. Börner, and S. M. Stigler, “Scientific progress despite irreproducibility: A seeming paradox,” *Proceedings of the National Academy of Sciences*, vol. 115, no. 11, pp. 2632–2639, Mar. 13, 2018, ISSN: 0027-8424, 1091-6490. DOI: [10.1073/pnas.1711786114](https://doi.org/10.1073/pnas.1711786114). [Online]. Available: <http://www.pnas.org/lookup/doi/10.1073/pnas.1711786114>.
- [20] R. Nuzzo, “Scientific method: Statistical errors,” *Nature News*, vol. 506, no. 7487, p. 150, Feb. 13, 2014, Section: News Feature. DOI: [10.1038/506150a](https://doi.org/10.1038/506150a). [Online]. Available: <http://www.nature.com/news/scientific-method-statistical-errors-1.14700>.
- [21] J. T. Leek and R. D. Peng, “Opinion: Reproducible research can still be wrong: Adopting a prevention approach,” *Proceedings of the National Academy of Sciences*, vol. 112, no. 6, pp. 1645–1646, Feb. 10, 2015, ISSN: 0027-8424, 1091-6490. DOI: [10.1073/pnas.1421412111](https://doi.org/10.1073/pnas.1421412111). [Online]. Available: <http://www.pnas.org/lookup/doi/10.1073/pnas.1421412111>.
- [22] R. D. Peng and S. C. Hicks, “Reproducible research: A retrospective,” *arXiv:2007.12210 [stat]*, Jul. 23, 2020. arXiv: [2007.12210](https://arxiv.org/abs/2007.12210). [Online]. Available: <http://arxiv.org/abs/2007.12210>.
- [23] C. Drummond, “Replicability is not reproducibility: Nor is it good science,” 2009. DOI: [10.1.1.149.5692](https://doi.org/10.1.1.149.5692). [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.5692&rep=rep1&type=pdf>.
- [24] C. L. Borgman, “The conundrum of sharing research data,” *Journal of the American Society for Information Science and Technology*, vol. 63, no. 6, pp. 1059–1078, Jun. 2012, ISSN: 15322882. DOI: [10.1002/asi.22634](https://doi.org/10.1002/asi.22634). [Online]. Available: <http://doi.wiley.com/10.1002/asi.22634>.
- [25] P. J. McMurdie and S. Holmes, “Phyloseq: An r package for reproducible interactive analysis and graphics of microbiome census data,” *PLoS ONE*, vol. 8, no. 4, M. Watson, Ed., e61217, Apr. 22, 2013, ISSN: 1932-6203. DOI: [10.1371/journal.pone.0061217](https://doi.org/10.1371/journal.pone.0061217). [Online]. Available: <https://dx.plos.org/10.1371/journal.pone.0061217>.
- [26] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, “Ten simple rules for reproducible computational research,” *PLoS Computational Biology*, vol. 9, no. 10, P. E. Bourne, Ed., e1003285, Oct. 24, 2013, ISSN: 1553-7358. DOI: [10.1371/journal.pcbi.1003285](https://doi.org/10.1371/journal.pcbi.1003285). [Online]. Available: <https://dx.plos.org/10.1371/journal.pcbi.1003285>.
- [27] K. Ram, “Git can facilitate greater reproducibility and increased transparency in science,” *Source Code for Biology and Medicine*, vol. 8, no. 1, p. 7, Dec. 2013, ISSN: 1751-0473. DOI: [10.1186/1751-0473-8-7](https://doi.org/10.1186/1751-0473-8-7). [Online]. Available: <https://scfbm.biomedcentral.com/articles/10.1186/1751-0473-8-7>.
- [28] C. Boettiger, “An introduction to docker for reproducible research,” *ACM SIGOPS Operating Systems Review*, vol. 49, no. 1, pp. 71–79, Jan. 20, 2015, ISSN: 0163-5980. DOI: [10.1145/2723872.2723882](https://doi.org/10.1145/2723872.2723882). [Online]. Available: <https://dl.acm.org/doi/10.1145/2723872.2723882>.
- [29] R. J. Nemiroff and J. Wallin. (1999). “Astrophysics source code library,” ASCL.net - Welcome to the ASCL, [Online]. Available: <http://ascl.net/>.
- [30] P. Ginsparg. (Aug. 14, 1991). “About arXiv | arXiv e-print repository,” [Online]. Available: <https://arxiv.org/about>.

- [31] A. M. Smith, K. E. Niemeyer, D. S. Katz, L. A. Barba, G. Githinji, M. Gymrek, K. D. Huff, C. R. Madan, A. Cabunoc Mayes, K. M. Moerman, P. Prins, K. Ram, A. Rokem, T. K. Teal, R. Valls Guimera, and J. T. Vanderplas, “Journal of open source software (JOSS): Design and first-year review,” *PeerJ Computer Science*, vol. 4, e147, Feb. 2018, ISSN: 2376-5992. DOI: [10.7717/peerj-cs.147](https://doi.org/10.7717/peerj-cs.147). [Online]. Available: <https://doi.org/10.7717/peerj-cs.147>.
- [32] F. C. Y. Benureau and N. P. Rougier, “Re-run, repeat, reproduce, reuse, replicate: Transforming code into scientific contributions,” *Frontiers in Neuroinformatics*, vol. 11, p. 69, 2018, ISSN: 1662-5196. DOI: [10.3389/fninf.2017.00069](https://doi.org/10.3389/fninf.2017.00069). [Online]. Available: <https://www.frontiersin.org/article/10.3389/fninf.2017.00069>.
- [33] L. A. Barba, “Terminologies for reproducible research,” *arXiv e-prints*, arXiv:1802.03311, Feb. 2018.
- [34] M. R. Munafó, B. A. Nosek, D. V. M. Bishop, K. S. Button, C. D. Chambers, N. Percie du Sert, U. Simonsohn, E.-J. Wagenmakers, J. J. Ware, and J. P. A. Ioannidis, “A manifesto for reproducible science,” *Nature Human Behaviour*, vol. 1, no. 1, p. 0021, Jan. 10, 2017, ISSN: 2397-3374. DOI: [10.1038/s41562-016-0021](https://doi.org/10.1038/s41562-016-0021). [Online]. Available: <https://doi.org/10.1038/s41562-016-0021>.
- [35] J. P. A. Ioannidis, “Meta-research: Why research on research matters,” *PLOS Biology*, vol. 16, no. 3, e2005468, Mar. 13, 2018, ISSN: 1545-7885. DOI: [10.1371/journal.pbio.2005468](https://doi.org/10.1371/journal.pbio.2005468). [Online]. Available: <https://dx.plos.org/10.1371/journal.pbio.2005468>.
- [36] S. B. Nissen, T. Magidson, K. Gross, and C. T. Bergstrom, “Research: Publication bias and the canonization of false facts,” *eLife*, vol. 5, P. Rodgers, Ed., e21451, Dec. 2016, Publisher: eLife Sciences Publications, Ltd, ISSN: 2050-084X. DOI: [10.7554/eLife.21451](https://doi.org/10.7554/eLife.21451). [Online]. Available: <https://doi.org/10.7554/eLife.21451>.
- [37] R. Botvinik-Nezer, F. Holzmeister, C. F. Camerer, A. Dreber, J. Huber, M. Johannesson, M. Kirchler, R. Iwanir, J. A. Mumford, R. A. Adcock, P. Avesani, B. M. Baczkowski, A. Bajracharya, L. Bakst, S. Ball, M. Barilari, N. Bault, D. Beaton, J. Beitner, R. G. Benoit, R. M. W. J. Berkers, J. P. Bhanji, B. B. Biswal, S. Bobadilla-Suarez, T. Bortolini, K. L. Bottenhorn, A. Bowring, S. Braem, H. R. Brooks, E. G. Brudner, C. B. Calderon, J. A. Camilleri, J. J. Castrellon, L. Cecchetti, E. C. Cieslik, Z. J. Cole, O. Collignon, R. W. Cox, W. A. Cunningham, S. Czoschke, K. Dadi, C. P. Davis, A. D. Luca, M. R. Delgado, L. Demetriou, J. B. Dennison, X. Di, E. W. Dickie, E. Dobryakova, C. L. Donnat, J. Dukart, N. W. Duncan, J. Durnez, A. Eed, S. B. Eickhoff, A. Erhart, L. Fontanesi, G. M. Fricke, S. Fu, A. Galván, R. Gau, S. Genon, T. Glatard, E. Glerean, J. J. Goeman, S. A. E. Golowin, C. González-García, K. J. Gorgolewski, C. L. Grady, M. A. Green, J. a. F. Guassi Moreira, O. Guest, S. Hakimi, J. P. Hamilton, R. Hancock, G. Handjaras, B. B. Harry, C. Hawco, P. Herholz, G. Herman, S. Heunis, F. Hoffstaedter, J. Hogeveen, S. Holmes, C.-P. Hu, S. A. Huettel, M. E. Hughes, V. Iacovella, A. D. Iordan, P. M. Isager, A. I. Isik, A. Jahn, M. R. Johnson, T. Johnstone, M. J. E. Joseph, A. C. Juliano, J. W. Kable, M. Kassinosopoulos, C. Koba, X.-Z. Kong, T. R. Kosciuk, N. E. Kucukboyaci, B. A. Kuhl, S. Kupek, A. R. Laird, C. Lamm, R. Langner, N. Lauharatanahirun, H. Lee, S. Lee, A. Leemans, A. Leo, E. Lesage, F. Li, M. Y. C. Li, P. C. Lim, E. N. Lintz, S. W. Liphardt, A. B.

- Losecaat Vermeer, B. C. Love, M. L. Mack, N. Malpica, T. Marins, C. Maumet, K. McDonald, J. T. McGuire, H. Melero, A. S. Méndez Leal, B. Meyer, K. N. Meyer, G. Mihai, G. D. Mitsis, J. Moll, D. M. Nielson, G. Nilsson, M. P. Notter, E. Olivetti, A. I. Onicas, P. Papale, K. R. Patil, J. E. Peelle, A. Pérez, D. Pischedda, J.-B. Poline, Y. Prystauka, S. Ray, P. A. Reuter-Lorenz, R. C. Reynolds, E. Ricciardi, J. R. Rieck, A. M. Rodriguez-Thompson, A. Romyn, T. Salo, G. R. Samanez-Larkin, E. Sanz-Morales, M. L. Schlichting, D. H. Schultz, Q. Shen, M. A. Sheridan, J. A. Silvers, K. Skagerlund, A. Smith, D. V. Smith, P. Sokol-Hessner, S. R. Steinkamp, S. M. Tashjian, B. Thirion, J. N. Thorp, G. Tinghög, L. Tisdall, S. H. Tompson, C. Toro-Serey, J. J. Torre Tresols, L. Tozzi, V. Truong, L. Turella, A. E. van t Veer, T. Verguts, J. M. Vettel, S. Vijayarajah, K. Vo, M. B. Wall, W. D. Weeda, S. Weis, D. J. White, D. Wisniewski, A. Xifra-Porxas, E. A. Yearling, S. Yoon, R. Yuan, K. S. L. Yuen, L. Zhang, X. Zhang, J. E. Zosky, T. E. Nichols, R. A. Poldrack, and T. Schonberg, “Variability in the analysis of a single neuroimaging dataset by many teams,” *Nature*, vol. 582, no. 7810, pp. 84–88, Jun. 2020, Number: 7810 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: [10.1038/s41586-020-2314-9](https://doi.org/10.1038/s41586-020-2314-9). [Online]. Available: <https://www.nature.com/articles/s41586-020-2314-9>.
- [38] B. A. Nosek, C. R. Ebersole, A. C. DeHaven, and D. T. Mellor, “The preregistration revolution,” *Proceedings of the National Academy of Sciences*, vol. 115, no. 11, pp. 2600–2606, 2018, Publisher: National Academy of Sciences .eprint: <https://www.pnas.org/content/115/11/2600.full.pdf>, ISSN: 0027-8424. DOI: [10.1073/pnas.1708274114](https://doi.org/10.1073/pnas.1708274114). [Online]. Available: <https://www.pnas.org/content/115/11/2600>.
- [39] I. V. Pasquetto, B. M. Randles, and C. L. Borgman, “On the reuse of scientific data,” *Data Science Journal*, vol. 16, p. 8, Mar. 22, 2017, ISSN: 1683-1470. DOI: [10.5334/dsj-2017-008](https://doi.org/10.5334/dsj-2017-008). [Online]. Available: <http://datascience.codata.org/articles/10.5334/dsj-2017-008/>.
- [40] E. Bellini, “A blockchain based trusted persistent identifier system for big data in science,” *Foundations of Computing and Decision Sciences*, vol. 44, no. 4, pp. 351–377, Dec. 1, 2019, ISSN: 2300-3405. DOI: [10.2478/fcds-2019-0018](https://doi.org/10.2478/fcds-2019-0018). [Online]. Available: <https://content.sciendo.com/doi/10.2478/fcds-2019-0018>.
- [41] A.-L. Lamprecht, L. Garcia, M. Kuzak, C. Martinez, R. Arcila, E. Martin Del Pico, V. Dominguez Del Angel, S. van de Sandt, J. Ison, P. A. Martinez, P. McQuilton, A. Valencia, J. Harrow, F. Psomopoulos, J. L. Gelpi, N. Chue Hong, C. Goble, and S. Capella-Gutierrez, “Towards FAIR principles for research software,” *Data Science*, vol. 3, no. 1, pp. 37–59, 2020, Publisher: IOS Press, ISSN: 2451-8492. DOI: [10.3233/DS-190026](https://doi.org/10.3233/DS-190026).
- [42] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. t Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, “The FAIR guiding principles for scientific data management

- and stewardship,” *Scientific Data*, vol. 3, no. 1, p. 160018, Mar. 15, 2016, ISSN: 2052-4463. DOI: [10.1038/sdata.2016.18](https://doi.org/10.1038/sdata.2016.18). [Online]. Available: <https://doi.org/10.1038/sdata.2016.18>.
- [43] M. Konkol, D. Nüst, and L. Goulier, “Publishing computational research - a review of infrastructures for reproducible and transparent scholarly communication,” *Research Integrity and Peer Review*, vol. 5, no. 1, Jul. 2020, Publisher: Springer Science and Business Media LLC, ISSN: 2058-8615. DOI: [10.1186/s41073-020-00095-y](https://doi.org/10.1186/s41073-020-00095-y). [Online]. Available: <http://dx.doi.org/10.1186/s41073-020-00095-y>.
- [44] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, “Scientific workflows: Past, present and future,” *Future Generation Computer Systems*, vol. 75, pp. 216–227, Oct. 1, 2017, ISSN: 0167-739X. DOI: [10.1016/j.future.2017.05.041](https://doi.org/10.1016/j.future.2017.05.041). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X17311202>.
- [45] B. K. Beaulieu-Jones and C. S. Greene, “Reproducibility of computational workflows is automated using continuous analysis,” *Nature Biotechnology*, vol. 35, no. 4, pp. 342–346, Apr. 2017, ISSN: 1087-0156, 1546-1696. DOI: [10.1038/nbt.3780](https://doi.org/10.1038/nbt.3780). [Online]. Available: <http://www.nature.com/articles/nbt.3780>.
- [46] S. S. Feger, “Interactive tools for reproducible science – understanding, supporting, and motivating reproducible science practices,” Nov. 10, 2020. [Online]. Available: <https://arxiv.org/abs/2012.02570v1>.
- [47] X. Chen, S. Dallmeier-Tiessen, R. Dasler, S. Feger, P. Fokianos, J. B. Gonzalez, H. Hirvonsalo, D. Kousidis, A. Lavasa, S. Mele, D. R. Rodriguez, T. Šimko, T. Smith, A. Trisovic, A. Trzcinska, I. Tsanaktsidis, M. Zimmermann, K. Cranmer, L. Heinrich, G. Watts, M. Hildreth, L. Lloret Iglesias, K. Lassila-Perini, and S. Neubert, “Open is not enough,” *Nature Physics*, vol. 15, no. 2, pp. 113–119, Feb. 2019, ISSN: 1745-2473, 1745-2481. DOI: [10.1038/s41567-018-0342-2](https://doi.org/10.1038/s41567-018-0342-2). [Online]. Available: <http://www.nature.com/articles/s41567-018-0342-2>.
- [48] O. E. Gundersen, “The fundamental principles of reproducibility,” *arXiv:2011.10098 [cs]*, Nov. 19, 2020. arXiv: [2011.10098](https://arxiv.org/abs/2011.10098). [Online]. Available: <http://arxiv.org/abs/2011.10098>.
- [49] P. Godfrey-Smith, *Theory and reality : an introduction to the philosophy of science*. Chicago: Univ. of Chicago Press, 2008, ISBN: 0-226-30062-5 0-226-30063-3 978-0-226-30062-7 978-0-226-30063-4.
- [50] N. K. Smith, *Immanuel Kant’s critique of pure reason*. Read Books Ltd, 2011.
- [51] W. V. Quine, “Main trends in recent philosophy: Two dogmas of empiricism,” *The philosophical review*, pp. 20–43, 1951, Publisher: JSTOR.
- [52] N. Goodman, *Fact, fiction, and forecast*. Harvard University Press, 1983.
- [53] K. Popper, *Conjectures and refutations: The growth of scientific knowledge*. routledge, 2014.
- [54] T. S. Kuhn, *The structure of scientific revolutions*. University of Chicago press, 2012.
- [55] P. Hoyningen-Huene, *Reconstructing scientific revolutions: Thomas S. Kuhn’s philosophy of science*. University of Chicago Press, 1993.

- [56] P. Kitcher, *The advancement of science: Science without legend, objectivity without illusions*. Oxford University Press on Demand, 1995.
- [57] I. Lakatos, “Falsification and the methodology of scientific research programmes,” in *Can theories be refuted?* Springer, 1976, pp. 205–259.
- [58] L. Laudan, *Progress and its problems: Towards a theory of scientific growth*. Univ of California Press, 1978, vol. 282.
- [59] P. Feyerabend and others, *Against method*. Verso, 1993.
- [60] R. K. Merton, *The sociology of science: Theoretical and empirical investigations*. University of Chicago press, 1973.
- [61] B. Barnes and D. Bloor, “Relativism, rationalism and the sociology of knowledge,” *Rationality and relativism*, pp. 21–47, 1982, Publisher: Oxford.
- [62] S. Shapin and S. Schaffer, *Leviathan and the air-pump: Hobbes, Boyle, and the experimental life*. Princeton University Press, 2011, vol. 109.
- [63] B. Latour and S. Woolgar, *Laboratory life: The construction of scientific facts*. Princeton University Press, 2013.
- [64] W. V. Quine and others, *Epistemology naturalized*. Herder, 1968.
- [65] D. L. Hull, *Science as a process: an evolutionary account of the social and conceptual development of science*. University of Chicago Press, 2010, ISBN: 978-0-226-36049-2.
- [66] W. Talbott, “Bayesian epistemology,” in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Winter 2016, Metaphysics Research Lab, Stanford University, 2016. [Online]. Available: <https://plato.stanford.edu/archives/win2016/entries/epistemology-bayesian/>.
- [67] (). “The SKA project,” Square Kilometre Array, [Online]. Available: <https://www.skatelescope.org/the-ska-project/>.
- [68] R. Braun, T. L. Bourke, J. A. Green, E. Keane, and J. Wagg, “Advancing astrophysics with the square kilometre array,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015, p. 174. DOI: [10.22323/1.215.0174](https://pos.sissa.it/215/174). [Online]. Available: <https://pos.sissa.it/215/174>.
- [69] G. Mellema, L. Koopmans, H. Shukla, K. K. Datta, A. Mesinger, and S. Majumdar, “HI tomographic imaging of the cosmic dawn and epoch of reionization with SKA,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015, p. 010. DOI: [10.22323/1.215.0010](https://pos.sissa.it/215/010). [Online]. Available: <https://pos.sissa.it/215/010>.
- [70] L. Staveley-Smith and T. Oosterloo, “HI science with the square kilometre array,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015, p. 167. DOI: [10.22323/1.215.0167](https://pos.sissa.it/215/167). [Online]. Available: <https://pos.sissa.it/215/167>.
- [71] M. Kramer and B. Stappers, “Pulsar science with the SKA,” *arXiv preprint arXiv:1507.04423*, 2015.

- [72] J. P. Macquart, E. Keane, K. Grainge, M. McQuinn, R. Fender, J. Hessels, A. Deller, R. Bhat, R. Breton, S. Chatterjee, C. Law, D. Lorimer, E. O. Ofek, M. Pietka, L. Spitler, B. Stappers, and C. Trott, “Fast transients at cosmological distances with the SKA,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015, p. 055. DOI: [10.22323/1.215.0055](https://doi.org/10.22323/1.215.0055). [Online]. Available: <https://pos.sissa.it/215/055>.
- [73] M. G. Hoare, L. Perez, T. L. Bourke, L. Testi, I. Jimenez-Serra, P. Zarka, P. V. Siemion, H. J. v. Langeveld, L. Lolnard, G. Anglada, A. Belloche, P. Bergman, R. Booth, P. Caselli, C. J. Chandler, C. Codella, G. Hallinan, J. Lazlo, I. S. Morrison, L. Podio, A. Remijan, and J. Tarter, “The cradle of life and the SKA,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015. DOI: [10.22323/1.215.0126](https://doi.org/10.22323/1.215.0126). [Online]. Available: <https://pos.sissa.it/215/126>.
- [74] P. Quinn, T. Axelrod, I. Bird, R. Dodson, A. Szalay, and A. Wicenec, “Delivering SKA science,” in *Proceedings of Advancing Astrophysics with the Square Kilometre Array PoS(AASKA14)*, Giardini Naxos, Italy: Sissa Medialab, May 29, 2015, p. 147. DOI: [10.22323/1.215.0147](https://doi.org/10.22323/1.215.0147). [Online]. Available: <https://pos.sissa.it/215/147>.
- [75] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and others, “Taverna: A tool for the composition and enactment of bioinformatics workflows,” *Bioinformatics*, vol. 20, no. 17, pp. 3045–3054, Jul. 2004, Publisher: Oxford University Press. DOI: [10.1093/bioinformatics/bth361](https://doi.org/10.1093/bioinformatics/bth361). [Online]. Available: <https://doi.org/10.1093/bioinformatics/bth361>.
- [76] P. Järveläinen, V. Savolainen, T. Oittinen, S. Maisala, M. H. Ullgrén, and others, “Using ESO reflex with web services,” in *Astronomical Data Analysis Software and Systems XVII*, vol. 394, 2008, p. 273, ISBN: 978-1-58381-658-5. [Online]. Available: <http://www.aspbbooks.org/publications/394/273.pdf>.
- [77] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, and C. Wroe, “Taverna: Lessons in creating a workflow environment for the life sciences: Research articles,” *Concurr. Comput. : Pract. Exper.*, vol. 18, no. 10, pp. 1067–1100, Aug. 2006, Place: GBR Publisher: John Wiley and Sons Ltd., ISSN: 1532-0626.
- [78] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, “Taverna: A tool for building and running workflows of services,” *Nucleic acids research*, vol. 34, W729–732, Web Server issue Jul. 1, 2006, ISSN: 1362-4962 0305-1048. DOI: [10.1093/nar/gkl320](https://doi.org/10.1093/nar/gkl320). [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1538887/>.
- [79] P. Missier, S. Soiland-Reyes, S. Owen, W. Tan, A. Nenadic, I. Dunlop, A. Williams, T. Oinn, and C. Goble, “Taverna, reloaded,” in *Scientific and Statistical Database Management*, M. Gertz and B. Ludäscher, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2010, pp. 471–481, ISBN: 978-3-642-13818-8. DOI: [10.1007/978-3-642-13818-8_33](https://doi.org/10.1007/978-3-642-13818-8_33).

- [80] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S. Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M. P. Balcazar Vargas, S. Sufi, and C. Goble, “The taverna workflow suite: Designing and executing workflows of web services on the desktop, web or in the cloud,” *Nucleic Acids Research*, vol. 41, W557–W561, Web Server issue Jul. 2013, ISSN: 0305-1048. DOI: [10.1093/nar/gkt328](https://doi.org/10.1093/nar/gkt328). [Online]. Available: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3692062/>.
- [81] B. Giardine, C. Riemer, R. C. Hardison, R. Burhans, L. Elnitski, P. Shah, Y. Zhang, D. Blankenberg, I. Albert, J. Taylor, W. Miller, W. J. Kent, and A. Nekrutenko, “Galaxy: A platform for interactive large-scale genome analysis,” *Genome Research*, vol. 15, no. 10, pp. 1451–1455, Oct. 1, 2005, Company: Cold Spring Harbor Laboratory Press Distributor: Cold Spring Harbor Laboratory Press Institution: Cold Spring Harbor Laboratory Press Label: Cold Spring Harbor Laboratory Press Publisher: Cold Spring Harbor Lab, ISSN: 1088-9051, 1549-5469. DOI: [10.1101/gr.4086505](https://doi.org/10.1101/gr.4086505). [Online]. Available: <http://genome.cshlp.org/content/15/10/1451>.
- [82] J. Goecks, A. Nekrutenko, J. Taylor, and The Galaxy Team, “Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences,” *Genome Biology*, vol. 11, no. 8, R86, Aug. 25, 2010, ISSN: 1474-760X. DOI: [10.1186/gb-2010-11-8-r86](https://doi.org/10.1186/gb-2010-11-8-r86). [Online]. Available: <https://doi.org/10.1186/gb-2010-11-8-r86>.
- [83] E. Afgan, D. Baker, B. Batut, M. van denBeek, D. Bouvier, M. ech, J. Chilton, D. Clements, N. Coraor, B. A. Grüning, A. Guerler, J. Hillman-Jackson, S. Hiltmann, V. Jalili, H. Rasche, N. Soranzo, J. Goecks, J. Taylor, A. Nekrutenko, and D. Blankenberg, “The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update,” *Nucleic Acids Research*, vol. 46, W537–W544, W1 Jul. 2, 2018, Publisher: Oxford Academic, ISSN: 0305-1048. DOI: [10.1093/nar/gky379](https://doi.org/10.1093/nar/gky379). [Online]. Available: <https://academic.oup.com/nar/article/46/W1/W537/5001157>.
- [84] F. Bartusch, M. Hanussek, and J. Krüger, “Containerization of galaxy workflows increases reproducibility,” Aug. 14, 2018, Accepted: 2018-08-22T13:34:42Z Publisher: Universität Tübingen. DOI: [10.15496/publikation-25200](https://doi.org/10.15496/publikation-25200). [Online]. Available: <https://publikationen.uni-tuebingen.de/xmlui/handle/10900/83810>.
- [85] T. Fahringer, A. Jugravu, S. Pillana, R. Prodan, C. Seragiotto Jr, and H.-L. Truong, “ASKALON: A tool set for cluster and grid computing,” *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2, pp. 143–169, 2005. DOI: <https://doi.org/10.1002/cpe.929>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.929>.
- [86] T. Fahringer, J. Qin, and S. Hainzer, “Specification of grid workflow applications with AGWL: An abstract grid workflow language,” in *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, vol. 2, 2005, 676–685 Vol. 2. DOI: [10.1109/CCGRID.2005.1558629](https://doi.org/10.1109/CCGRID.2005.1558629).
- [87] T. Fahringer, S. Pillana, and J. Testori, “Teuta: Tool support for performance modeling of distributed and parallel applications,” in *Computational Science - ICCS 2004*, M. Bubak,

- G. D. van Albada, P. M. A. Sloot, and J. Dongarra, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 456–463, ISBN: 978-3-540-24688-6.
- [88] J. Qin, T. Fahringer, and S. Pillana, “UML based grid workflow modeling under ASKALON,” in *Proceedings of 6th Austrian-Hungarian Workshop on Distributed and Parallel Systems*, Innsbruck, Austria: Springer-Verlag, Sep. 21, 2006.
- [89] J. Qin and T. Fahringer, “Advanced data flow support for scientific grid workflow applications,” in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC07)*, Reno, NV, USA: IEEE Computer Society Press, Nov. 10, 2007. DOI: [10.1145/1362622.1362679](https://doi.org/10.1145/1362622.1362679).
- [90] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, “Pegasus: A framework for mapping complex scientific workflows onto distributed systems,” *Sci. Program.*, vol. 13, no. 3, pp. 219–237, Jul. 2005, Place: NLD Publisher: IOS Press. DOI: [10.1155/2005/128026](https://doi.org/10.1155/2005/128026). [Online]. Available: <https://doi.org/10.1155/2005/128026>.
- [91] J. Yu and R. Buyya, “A taxonomy of workflow management systems for grid computing,” *Journal of Grid Computing*, vol. 3, no. 3, pp. 171–200, Sep. 1, 2005, ISSN: 1572-9184. DOI: [10.1007/s10723-005-9010-8](https://doi.org/10.1007/s10723-005-9010-8). [Online]. Available: <https://doi.org/10.1007/s10723-005-9010-8>.
- [92] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, “Pegasus, a workflow management system for science automation,” *Future Generation Computer Systems*, vol. 46, pp. 17–35, May 2015, ISSN: 0167739X. DOI: [10.1016/j.future.2014.10.008](https://doi.org/10.1016/j.future.2014.10.008). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167739X14002015>.
- [93] E. Larssonneur, J. Mercier, N. Wiart, E. L. Floch, O. Delhomme, and V. Meyer, “Evaluating workflow management systems: A bioinformatics use case,” in *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, Dec. 2018, pp. 2773–2775. DOI: [10.1109/BIBM.2018.8621141](https://doi.org/10.1109/BIBM.2018.8621141).
- [94] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E. A. Lee, J. Tao, and Y. Zhao, “Scientific workflow management and the kepler system,” *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006, ISSN: 1532-0634. DOI: <https://doi.org/10.1002/cpe.994>. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.994>.
- [95] I. Altintas, O. Barney, and E. Jaeger-Frank, “Provenance collection support in the kepler scientific workflow system,” in *Provenance and Annotation of Data*, L. Moreau and I. Foster, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2006, pp. 118–132, ISBN: 978-3-540-46303-0. DOI: [10.1007/11890850_14](https://doi.org/10.1007/11890850_14).
- [96] E. Deelman, “Looking into the future of workflows: The challenges ahead,” in *Workflows for e-Science: Scientific Workflows for Grids*, I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds., London: Springer, 2007, pp. 475–481, ISBN: 978-1-84628-757-2. DOI: [10.1007/978-1-84628-757-2_28](https://doi.org/10.1007/978-1-84628-757-2_28). [Online]. Available: https://doi.org/10.1007/978-1-84628-757-2_28.

- [97] S. Bowers, T. McPhillips, S. Riddle, M. K. Anand, and B. Ludäscher, “Kepler/pPOD: Scientific workflow and provenance support for assembling the tree of life,” in *Provenance and Annotation of Data and Processes*, J. Freire, D. Koop, and L. Moreau, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2008, pp. 70–77, ISBN: 978-3-540-89965-5. DOI: [10.1007/978-3-540-89965-5_9](https://doi.org/10.1007/978-3-540-89965-5_9).
- [98] I. Taylor, M. Shields, I. Wang, and A. Harrison, “The triana workflow environment: Architecture and applications,” in *Workflows for e-Science: Scientific Workflows for Grids*, I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Eds., London: Springer, 2007, pp. 320–339, ISBN: 978-1-84628-757-2. DOI: [10.1007/978-1-84628-757-2_20](https://doi.org/10.1007/978-1-84628-757-2_20). [Online]. Available: https://doi.org/10.1007/978-1-84628-757-2_20.
- [99] M. R. Berthold, N. Cebon, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel, “KNIME - the konstanz information miner: Version 2.0 and beyond,” *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 26–31, Nov. 2009, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 1931-0145. DOI: [10.1145/1656274.1656280](https://doi.org/10.1145/1656274.1656280). [Online]. Available: <https://doi.org/10.1145/1656274.1656280>.
- [100] T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec, “Flexible and efficient workflow deployment of data-intensive applications on grids with MOTEUR,” *The International Journal of High Performance Computing Applications*, vol. 22, no. 3, pp. 347–360, 2008. DOI: [10.1177/1094342008096067](https://doi.org/10.1177/1094342008096067). [Online]. Available: <https://doi.org/10.1177/1094342008096067>.
- [101] J. Montagnat, B. Isnard, T. Glatard, K. Maheshwari, and M. Blay-Fornarino, “A data-driven workflow language for grids based on array programming principles,” in *International conference on High Performance Computing, networking, storage and analysis (SC09)*, Portland, United States: ACM, Nov. 2009, pp. 1–10. DOI: [10.1145/1645164.1645171](https://doi.org/10.1145/1645164.1645171). [Online]. Available: <https://hal.archives-ouvertes.fr/hal-00677806>.
- [102] J. R. Balderrama, T. T. Huu, and J. Montagnat, “Scalable and resilient workflow executions on production distributed computing infrastructures,” in *2012 11th International Symposium on Parallel and Distributed Computing*, 2012, pp. 119–126. DOI: [10.1109/ISPDC.2012.24](https://doi.org/10.1109/ISPDC.2012.24).
- [103] D. Rogers, I. Harvey, T. T. Huu, K. Evans, T. Glatard, I. Kallel, I. Taylor, J. Montagnat, A. Jones, and A. Harrison, “Bundle and pool architecture for multi-language, robust, scalable workflow executions,” *Journal of Grid Computing*, vol. 11, no. 3, pp. 457–480, Sep. 1, 2013, ISSN: 1572-9184. DOI: [10.1007/s10723-013-9267-2](https://doi.org/10.1007/s10723-013-9267-2). [Online]. Available: <https://doi.org/10.1007/s10723-013-9267-2>.
- [104] K. Plankensteiner, R. Prodan, M. Janetschek, T. Fahringer, J. Montagnat, D. Rogers, I. Harvey, I. Taylor, A. Balaskó, and P. Kacsuk, “Fine-grain interoperability of scientific workflows in distributed computing infrastructures,” *Journal of Grid Computing*, vol. 11, no. 3, pp. 429–455, Sep. 1, 2013, ISSN: 1572-9184. DOI: [10.1007/s10723-013-9261-8](https://doi.org/10.1007/s10723-013-9261-8). [Online]. Available: <https://doi.org/10.1007/s10723-013-9261-8>.

- [105] P. Di Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, “Nextflow enables reproducible computational workflows,” *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, Apr. 1, 2017, <https://github.com/nextflow-io/nextflow><https://github.com/nextflow-io/awesome-nextflow>, ISSN: 1546-1696. DOI: [10.1038/nbt.3820](https://doi.org/10.1038/nbt.3820). [Online]. Available: <https://doi.org/10.1038/nbt.3820>.
- [106] T. Šimko, L. Heinrich, H. Hirvonsalo, D. Kousidis, and D. Rodríguez, “REANA: A system for reusable research data analyses,” *EPJ Web of Conferences*, vol. 214, A. Forti, L. Betev, M. Litmaath, O. Smirnova, and P. Hristov, Eds., p. 06 034, 2019, ISSN: 2100-014X. DOI: [10.1051/epjconf/201921406034](https://doi.org/10.1051/epjconf/201921406034). [Online]. Available: <https://www.epj-conferences.org/10.1051/epjconf/201921406034>.
- [107] C. Wu, R. Tobar, K. Vinsen, A. Wicenec, D. Pallot, B. Lao, R. Wang, T. An, M. Boulton, I. Cooper, R. Dodson, M. Dolensky, Y. Mei, and F. Wang, “DALiUGe: A graph execution framework for harnessing the astronomical data deluge,” *Astronomy and Computing*, vol. 20, pp. 1–15, Jul. 2017, ISSN: 22131337. DOI: [10.1016/j.ascom.2017.03.007](https://doi.org/10.1016/j.ascom.2017.03.007). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2213133716301214>.
- [108] S. B. Davidson and J. Freire, “Provenance and scientific workflows: Challenges and opportunities,” in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’08, event-place: Vancouver, Canada, New York, NY, USA: Association for Computing Machinery, 2008, pp. 1345–1350, ISBN: 978-1-60558-102-6. DOI: [10.1145/1376616.1376772](https://doi.org/10.1145/1376616.1376772). [Online]. Available: <https://doi.org/10.1145/1376616.1376772>.
- [109] Y. Gil, P. A. Gonzalez-Calero, J. Kim, J. Moody, and V. Ratnakar, “A semantic framework for automatic generation of computational workflows using distributed data and component catalogues,” *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 23, no. 4, pp. 389–467, 2011, Publisher: Taylor & Francis.
- [110] P. Kacsuk, T. Kiss, and G. Sipos, “Solving the grid interoperability problem by p-GRADE portal at workflow level,” *Future Generation Computer Systems*, vol. 24, no. 7, pp. 744–751, Jul. 1, 2008, ISSN: 0167-739X. DOI: [10.1016/j.future.2008.02.008](https://doi.org/10.1016/j.future.2008.02.008). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X08000113>.
- [111] K. Belhajjame, O. Corcho, D. Garijo, J. Zhao, P. Missier, D. R. Newman, R. Palma, S. Bechhofer, E. Garcia-Cuesta, J. M. Gomez-Perez, and others, “Workflow-centric research objects: A first class citizen in the scholarly discourse.,” in *SePublica@ ESWC*, 2012, pp. 1–12. [Online]. Available: <http://ceur-ws.org/Vol-903/paper-01.pdf>.
- [112] P. Missier, K. Belhajjame, and J. Cheney, “The w3c PROV family of specifications for modelling provenance metadata,” in *Proceedings of the 16th International Conference on Extending Database Technology*, ser. EDBT ’13, event-place: Genoa, Italy, New York, NY, USA: Association for Computing Machinery, 2013, pp. 773–776, ISBN: 978-1-4503-1597-5. DOI: [10.1145/2452376.2452478](https://doi.org/10.1145/2452376.2452478). [Online]. Available: <https://doi.org/10.1145/2452376.2452478>.

- [113] S. Cohen-Boulakia, K. Belhajjame, O. Collin, J. Chopard, C. Froidevaux, A. Gaignard, K. Hinsén, P. Larmande, Y. L. Bras, F. Lemoine, F. Mareuil, H. Ménager, C. Pradal, and C. Blanchet, “Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities,” *Future Generation Computer Systems*, vol. 75, pp. 284–298, 2017, ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2017.01.012>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X17300316>.
- [114] B. Pérez, J. Rubio, and C. Sáenz-Adán, “A systematic review of provenance systems,” *Knowledge and Information Systems*, vol. 57, no. 3, pp. 495–543, Dec. 1, 2018, ISSN: 0219-3116. DOI: [10.1007/s10115-018-1164-3](https://doi.org/10.1007/s10115-018-1164-3). [Online]. Available: <https://doi.org/10.1007/s10115-018-1164-3>.
- [115] C. Goble, S. Cohen-Boulakia, S. Soiland-Reyes, D. Garijo, Y. Gil, M. R. Crusoe, K. Peters, and D. Schober, “FAIR computational workflows,” *Data Intelligence*, vol. 2, no. 1, pp. 108–121, Nov. 1, 2019, Publisher: MIT Press. DOI: [10.1162/dint_a_00033](https://doi.org/10.1162/dint_a_00033). [Online]. Available: https://doi.org/10.1162/dint_a_00033.
- [116] R. Filgueira, A. Krause, M. Atkinson, I. Klampanos, A. Spinuso, and S. Sanchez-Exposito, “Dispel4py: An agile framework for data-intensive eScience,” in *2015 IEEE 11th International Conference on e-Science*, 2015, pp. 454–464. DOI: [10.1109/eScience.2015.40](https://doi.org/10.1109/eScience.2015.40).
- [117] J. E. Ruiz, J. Garrido, J. D. Santander-Vela, S. Sánchez-Expósito, and L. Verdes-Montenegro, “AstroTavernabuilding workflows with virtual observatory services,” *Astronomy and Computing*, vol. 7-8, pp. 3–11, 2014, ISSN: 2213-1337. DOI: <https://doi.org/10.1016/j.ascom.2014.09.002>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2213133714000419>.
- [118] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, “A characterization of workflow management systems for extreme-scale applications,” *Future Generation Computer Systems*, vol. 75, pp. 228–238, Oct. 1, 2017, ISSN: 0167-739X. DOI: [10.1016/j.future.2017.02.026](https://doi.org/10.1016/j.future.2017.02.026). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X17302510>.
- [119] A. Gaignard, J. Montagnat, B. Gibaud, G. Forestier, and T. Glatard, “Domain-specific summarization of life-science e-experiments from provenance traces,” *Journal of Web Semantics*, vol. 29, pp. 19–30, 2014, ISSN: 1570-8268. DOI: <https://doi.org/10.1016/j.websem.2014.07.001>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1570826814000493>.
- [120] A. Gaignard, H. Skaf-Molli, and K. Belhajjame, “Findable and reusable workflow data products: A genomic workflow case study,” *Semantic Web*, vol. 11, no. 5, pp. 751–763, Jan. 1, 2020, Publisher: IOS Press, ISSN: 1570-0844. DOI: [10.3233/SW-200374](https://doi.org/10.3233/SW-200374). [Online]. Available: <https://content.iospress.com/articles/semantic-web/sw200374>.
- [121] *The WorkflowHub*. [Online]. Available: <https://workflowhub.eu/> (visited on 10/08/2021).

- [122] T. Coleman, H. Casanova, L. Pottier, M. Kaushik, E. Deelman, and R. F. da Silva, “WfCommons: A framework for enabling scientific workflow research and development,” en, *Future Generation Computer Systems*, Oct. 2021, ISSN: 0167-739X. DOI: [10.1016/j.future.2021.09.043](https://doi.org/10.1016/j.future.2021.09.043). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X21003897> (visited on 10/08/2021).
- [123] D. Baker, M. v. d. Beek, D. Blankenberg, D. Bouvier, J. Chilton, N. Coraor, F. Coppens, I. Eguinoa, S. Gladman, B. Grüning, N. Keener, D. Larivière, A. Lonie, S. K. Pond, W. Maier, A. Nekrutenko, J. Taylor, and S. Weaver, “No more business as usual: Agile and effective responses to emerging pathogen threats require open data and open analytics,” *PLOS Pathogens*, vol. 16, no. 8, e1008643, Aug. 13, 2020, Publisher: Public Library of Science, ISSN: 1553-7374. DOI: [10.1371/journal.ppat.1008643](https://doi.org/10.1371/journal.ppat.1008643). [Online]. Available: <https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1008643>.
- [124] R. Celebi, J. R. Moreira, A. A. Hassan, S. Ayyar, L. Ridder, T. Kuhn, and M. Dumontier, “Towards FAIR protocols and workflows: The OpenPREDICT use case,” p. 29, 2020. DOI: <https://doi.org/10.7717/peerj-cs.281>. [Online]. Available: <https://peerj.com/articles/cs-281/>.
- [125] L. Lamport, “The implementation of reliable distributed multiprocess systems,” *Computer Networks (1976)*, vol. 2, no. 2, pp. 95–114, May 1978, ISSN: 03765075. DOI: [10.1016/0376-5075\(78\)90045-4](https://doi.org/10.1016/0376-5075(78)90045-4). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0376507578900454>.
- [126] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, Jul. 1, 1982, ISSN: 01640925. DOI: [10.1145/357172.357176](https://doi.org/10.1145/357172.357176). [Online]. Available: <http://portal.acm.org/citation.cfm?doid=357172.357176>.
- [127] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” *ACM Computing Surveys*, vol. 22, no. 4, pp. 299–319, Dec. 1, 1990, ISSN: 03600300. DOI: [10.1145/98163.98167](https://doi.org/10.1145/98163.98167). [Online]. Available: <http://portal.acm.org/citation.cfm?doid=98163.98167>.
- [128] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0004-5411. DOI: [10.1145/3149.214121](https://doi.org/10.1145/3149.214121). [Online]. Available: <https://doi.org/10.1145/3149.214121>.
- [129] L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, May 1998, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0734-2071. DOI: [10.1145/279227.279229](https://doi.org/10.1145/279227.279229). [Online]. Available: <https://doi.org/10.1145/279227.279229>.
- [130] M. Castro and B. Liskov, “Practical byzantine fault tolerance and proactive recovery,” *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, Nov. 2002, ISSN: 0734-2071, 1557-7333. DOI: [10.1145/571637.571640](https://doi.org/10.1145/571637.571640). [Online]. Available: <http://dl.acm.org/doi/10.1145/571637.571640>.

- [131] S. Gilbert and N. Lynch, “Brewer’s conjecture and the feasibility of consistent, available, partition-tolerant web services,” *ACM SIGACT News*, vol. 33, no. 2, p. 51, Jun. 1, 2002, ISSN: 01635700. DOI: [10.1145/564585.564601](https://doi.org/10.1145/564585.564601). [Online]. Available: <http://portal.acm.org/citation.cfm?doid=564585.564601>.
- [132] L. Lamport, “Lower bounds for asynchronous consensus,” *Distributed Computing*, vol. 19, no. 2, pp. 104–125, Oct. 1, 2006, ISSN: 1432-0452. DOI: [10.1007/s00446-006-0155-x](https://doi.org/10.1007/s00446-006-0155-x). [Online]. Available: <https://doi.org/10.1007/s00446-006-0155-x>.
- [133] B. Kemme and G. Alonso, “A new approach to developing and implementing eager database replication protocols,” *ACM Transactions on Database Systems*, vol. 25, no. 3, pp. 333–379, Sep. 1, 2000, ISSN: 03625915. DOI: [10.1145/363951.363955](https://doi.org/10.1145/363951.363955). [Online]. Available: <http://portal.acm.org/citation.cfm?doid=363951.363955>.
- [134] M. Burrows, “The chubby lock service for loosely-coupled distributed systems,” p. 16, Jan. 11, 2006. [Online]. Available: <https://static.googleusercontent.com/media/research.google.com/en//archive/chubby-osdi06.pdf>.
- [135] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” *ACM Transactions on Computer Systems*, vol. 26, no. 2, pp. 1–26, Jun. 1, 2008, ISSN: 07342071. DOI: [10.1145/1365815.1365816](https://doi.org/10.1145/1365815.1365816). [Online]. Available: <http://portal.acm.org/citation.cfm?doid=1365815.1365816>.
- [136] B. Kemme, R. Jiménez Peris, and M. Patiño Martínez, *Database replication*, ser. Synthesis lectures on data management 7. San Rafael, Calif.: Morgan & Claypool, May 6, 2010, OCLC: 732116595, ISBN: 978-1-60845-381-8 978-1-60845-382-5.
- [137] F. P. Junqueira, B. C. Reed, and M. Serafini, “Zab: High-performance broadcast for primary-backup systems,” in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, Hong Kong: IEEE, Jun. 2011, pp. 245–256, ISBN: 978-1-4244-9233-6 978-1-4244-9232-9 978-1-4244-9231-2. DOI: [10.1109/DSN.2011.5958223](https://doi.org/10.1109/DSN.2011.5958223). [Online]. Available: <http://ieeexplore.ieee.org/document/5958223/>.
- [138] P. J. Marandi, M. Primi, and F. Pedone, “High performance state-machine replication,” in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, Hong Kong, China: IEEE, Jun. 2011, pp. 454–465, ISBN: 978-1-4244-9232-9. DOI: [10.1109/DSN.2011.5958258](https://doi.org/10.1109/DSN.2011.5958258). [Online]. Available: <http://ieeexplore.ieee.org/document/5958258/>.
- [139] M. Kapritsos, Y. Wang, V. Quema, A. Clement, L. Alvisi, and M. Dahlin, “All about eve: Execute-verify replication for multi-core servers,” in *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI’12, event-place: Hollywood, CA, USA, USA: USENIX Association, 2012, pp. 237–250, ISBN: 978-1-931971-96-6.
- [140] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, and others, “Spanner: Googles globally distributed database,” *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, pp. 1–22, 2013, Publisher: ACM New York, NY, USA.

- [141] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” p. 9, Oct. 31, 2008.
- [142] C. Dwork and M. Naor, “Pricing via processing or combatting junk mail,” in *Advances in Cryptology CRYPTO 92*, E. F. Brickell, Ed., vol. 740, Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 139–147, ISBN: 978-3-540-57340-1. DOI: [10.1007/3-540-48071-4_10](https://doi.org/10.1007/3-540-48071-4_10). [Online]. Available: http://link.springer.com/10.1007/3-540-48071-4_10.
- [143] J. Waldo, “A hitchhiker’s guide to the blockchain universe,” *Communications of the ACM*, vol. 62, no. 3, pp. 38–42, Feb. 21, 2019, ISSN: 00010782. DOI: [10.1145/3303868](https://doi.org/10.1145/3303868). [Online]. Available: <http://dl.acm.org/citation.cfm?doid=3314328.3303868>.
- [144] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, “Zerocash: Decentralized anonymous payments from bitcoin,” in *2014 IEEE Symposium on Security and Privacy*, ISSN: 2375-1207, May 2014, pp. 459–474. DOI: [10.1109/SP.2014.36](https://doi.org/10.1109/SP.2014.36).
- [145] N. Szabo, “Smart contracts : Building blocks for digital markets,” Jan. 1, 1996.
- [146] V. Buterin and others, “A next-generation smart contract and decentralized application platform,” *white paper*, vol. 3, no. 37, 2014. [Online]. Available: https://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.
- [147] I. Sergey and A. Hobor, “A concurrent perspective on smart contracts,” *arXiv:1702.05511 [cs]*, Feb. 17, 2017. arXiv: [1702.05511](https://arxiv.org/abs/1702.05511). [Online]. Available: <http://arxiv.org/abs/1702.05511>.
- [148] N. Atzei, M. Bartoletti, and T. Cimoli, “A survey of attacks on ethereum smart contracts SoK,” in *Proceedings of the 6th International Conference on Principles of Security and Trust - Volume 10204*, Berlin, Heidelberg: Springer-Verlag, 2017, pp. 164–186, ISBN: 978-3-662-54454-9. DOI: [10.1007/978-3-662-54455-6_8](https://doi.org/10.1007/978-3-662-54455-6_8). [Online]. Available: https://doi.org/10.1007/978-3-662-54455-6_8.
- [149] *Slockit/DAO*, original-date: 2016-03-02T13:35:03Z, Mar. 4, 2020. [Online]. Available: <https://github.com/slockit/DAO>.
- [150] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in *2016 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA: IEEE, May 2016, pp. 839–858, ISBN: 978-1-5090-0824-7. DOI: [10.1109/SP.2016.55](https://doi.org/10.1109/SP.2016.55). [Online]. Available: <http://ieeexplore.ieee.org/document/7546538/>.
- [151] J. K. Kwon, “Tendermint : Consensus without mining,” Jan. 1, 2014. [Online]. Available: <https://tendermint.com/static/docs/tendermint.pdf>.
- [152] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *USENIX ATC ’14*, Philadelphia, PA, Jun. 19, 2014, pp. 305–319. [Online]. Available: <https://www.usenix.org/conference/atc14/technical-sessions/presentation/ongaro>.
- [153] I. Eyal, A. E. Gencer, E. G. Sirer, and R. Van Renesse, “Bitcoin-NG: A scalable blockchain protocol,” in *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, ser. NSDI’16, event-place: Santa Clara, CA, USA: USENIX Association, 2016, pp. 45–59, ISBN: 978-1-931971-29-4. [Online]. Available: <https://dl.acm.org/doi/abs/10.5555/2930611.2930615>.

- [154] M. Vukoli, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Open Problems in Network Security*, J. Camenisch and D. Kesdoan, Eds., vol. 9591, Cham: Springer International Publishing, Jan. 5, 2016, pp. 112–125, ISBN: 978-3-319-39027-7 978-3-319-39028-4. DOI: [10.1007/978-3-319-39028-4_9](https://doi.org/10.1007/978-3-319-39028-4_9). [Online]. Available: http://link.springer.com/10.1007/978-3-319-39028-4_9.
- [155] S. Liu, P. Viotti, C. Cachin, M. Vukolic, and V. Quema, "XFT: Practical fault tolerance beyond crashes," in *12th USENIX Symposium on Operating Systems Design and Implementation*, vol. 12, Savannah, GA, USA, Nov. 2, 2016, pp. 485–500. [Online]. Available: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/liu>.
- [156] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukoli, S. W. Cocco, and J. Yellick, "Hyperledger fabric: A distributed operating system for permissioned blockchains," *Proceedings of the Thirteenth EuroSys Conference on - EuroSys '18*, pp. 1–15, Jan. 30, 2018. DOI: [10.1145/3190508.3190538](https://doi.org/10.1145/3190508.3190538). arXiv: [1801.10228](https://arxiv.org/abs/1801.10228). [Online]. Available: <http://arxiv.org/abs/1801.10228>.
- [157] D. Di Francesco Maesa and P. Mori, "Blockchain 3.0 applications survey," *Journal of Parallel and Distributed Computing*, vol. 138, pp. 99–114, Apr. 2020, ISSN: 07437315. DOI: [10.1016/j.jpdc.2019.12.019](https://doi.org/10.1016/j.jpdc.2019.12.019). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0743731519308664>.
- [158] B. Biswas and R. Gupta, "Analysis of barriers to implement blockchain in industry and service sectors," *Computers & Industrial Engineering*, vol. 136, pp. 225–241, Oct. 2019, ISSN: 03608352. DOI: [10.1016/j.cie.2019.07.005](https://doi.org/10.1016/j.cie.2019.07.005). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0360835219303961>.
- [159] M. Janssen, V. Weerakkody, E. Ismagilova, U. Sivarajah, and Z. Irani, "A framework for analysing blockchain technology adoption: Integrating institutional, market and technical factors," *International Journal of Information Management*, vol. 50, pp. 302–309, Feb. 2020, ISSN: 02684012. DOI: [10.1016/j.ijinfomgt.2019.08.012](https://doi.org/10.1016/j.ijinfomgt.2019.08.012). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0268401219305067>.
- [160] M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, and M. H. Rehmani, "Applications of blockchains in the internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1676–1717, Dec. 18, 2018, ISSN: 1553-877X, 2373-745X. DOI: [10.1109/COMST.2018.2886932](https://doi.org/10.1109/COMST.2018.2886932). [Online]. Available: <https://ieeexplore.ieee.org/document/8580364/>.
- [161] J. Sengupta, S. Ruj, and S. Das Bit, "A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT," *Journal of Network and Computer Applications*, vol. 149, p. 102481, Jan. 2020, ISSN: 10848045. DOI: [10.1016/j.jnca.2019.102481](https://doi.org/10.1016/j.jnca.2019.102481). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1084804519303418>.

- [162] T. Bocek, B. B. Rodrigues, T. Strasser, and B. Stiller, “Blockchains everywhere - a use-case of blockchains in the pharma supply-chain,” in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, May 2017, pp. 772–777. DOI: [10.23919/INM.2017.7987376](https://doi.org/10.23919/INM.2017.7987376).
- [163] D. Bumblauskas, A. Mann, B. Dugan, and J. Rittmer, “A blockchain use case in food distribution: Do you know where your food has been?” *International Journal of Information Management*, vol. 52, no. 102008, p. 10, Oct. 2019, ISSN: 02684012. DOI: [10.1016/j.ijinfomgt.2019.09.004](https://doi.org/10.1016/j.ijinfomgt.2019.09.004). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S026840121930461X>.
- [164] A. Miglani, N. Kumar, V. Chamola, and S. Zeadally, “Blockchain for internet of energy management: Review, solutions, and challenges,” *Computer Communications*, vol. 151, pp. 395–418, Feb. 2020, ISSN: 01403664. DOI: [10.1016/j.comcom.2020.01.014](https://doi.org/10.1016/j.comcom.2020.01.014). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0140366419314951>.
- [165] E. Münsing, J. Mather, and S. Moura, “Blockchains for decentralized optimization of energy resources in microgrid networks,” in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, Aug. 2017, pp. 2164–2171. DOI: [10.1109/CCTA.2017.8062773](https://doi.org/10.1109/CCTA.2017.8062773).
- [166] X. Liang, S. Shetty, D. Tosh, C. Kamhoua, K. Kwiat, and L. Njilla, “ProvChain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability,” in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2017, pp. 468–477. DOI: [10.1109/CCGRID.2017.8](https://doi.org/10.1109/CCGRID.2017.8).
- [167] D. Karastoyanova and L. Stage, “Towards collaborative and reproducible scientific experiments on blockchain,” in *Advanced Information Systems Engineering Workshops*, R. Matulevicius and R. Dijkman, Eds., Cham: Springer International Publishing, 2018, pp. 144–149, ISBN: 978-3-319-92898-2.
- [168] J. Evermann and H. Kim, “Workflow management on BFT blockchains,” *arXiv:1905.12652 [cs]*, May 29, 2019. arXiv: [1905.12652](https://arxiv.org/abs/1905.12652). [Online]. Available: <http://arxiv.org/abs/1905.12652>.
- [169] P. Wei, D. Wang, Y. Zhao, S. K. S. Tyagi, and N. Kumar, “Blockchain data-based cloud data integrity protection mechanism,” *Future Generation Computer Systems*, vol. 102, pp. 902–911, Jan. 2020, ISSN: 0167739X. DOI: [10.1016/j.future.2019.09.028](https://doi.org/10.1016/j.future.2019.09.028). [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167739X19313494>.
- [170] G. Terstyanszky, T. Kukla, T. Kiss, P. Kacsuk, A. Balasko, and Z. Farkas, “Enabling scientific workflow sharing through coarse-grained interoperability,” *Future Generation Computer Systems*, Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing, vol. 37, pp. 46–59, Jul. 1, 2014, ISSN: 0167-739X. DOI: [10.1016/j.future.2014.02.016](https://doi.org/10.1016/j.future.2014.02.016). [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0167739X14000417>.
- [171] P. Amstutz, M. R. Crusoe, Neboja Tijani, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic, *Common Workflow Language, v1.0*. figshare, 2016. DOI: [10.6084/M9.FIGSHARE.3115156.V2](https://doi.org/10.6084/M9.FIGSHARE.3115156.V2). [Online]. Available: https://figshare.com/articles/dataset/Common_Workflow_Language_draft_3/3115156/2.

- [172] K. Cranmer and L. Heinrich, “Yadage and packtivity analysis preservation using parametrized workflows,” *Journal of Physics: Conference Series*, vol. 898, p. 102019, Oct. 2017, Publisher: IOP Publishing. DOI: [10.1088/1742-6596/898/10/102019](https://doi.org/10.1088/1742-6596/898/10/102019). [Online]. Available: <https://doi.org/10.1088/1742-6596/898/10/102019>.
- [173] *The P-Plan Ontology*. [Online]. Available: <http://vocab.linkeddata.es/p-plan/index.html> (visited on 10/11/2021).
- [174] T. Fahringer, R. Prodan, Rubing Duan, F. Nerieri, S. Podlipnig, Jun Qin, M. Siddiqui, Hong-Linh Truong, A. Villazon, and M. Wiczorek, “ASKALON: A grid application development and computing environment,” in *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.*, 2005, 10 pp.–. DOI: [10.1109/GRID.2005.1542733](https://doi.org/10.1109/GRID.2005.1542733).
- [175] D. Garijo, N. Villanueva-Rosales, and T. Kauppinen, “Editorial: Special issue on semantic eScience: Methods, tools and applications,” *Semantic Web*, vol. 11, no. 5, pp. 731–733, 2020, Publisher: IOS Press, ISSN: 2210-4968. DOI: [10.3233/SW-200380](https://doi.org/10.3233/SW-200380).
- [176] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. d. Bussche, “The open provenance model core specification (v1.1),” *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, 2011, ISSN: 0167-739X. DOI: <https://doi.org/10.1016/j.future.2010.07.005>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X10001275>.
- [177] P. A. Ewels, A. Peltzer, S. Fillinger, H. Patel, J. Alneberg, A. Wilm, M. U. Garcia, P. Di Tommaso, and S. Nahnsen, “The nf-core framework for community-curated bioinformatics pipelines,” in *Nature Biotechnology*, vol. 38, no. 3, pp. 276–278, Mar. 2020, ISSN: 1087-0156, 1546-1696. DOI: [10.1038/s41587-020-0439-x](https://doi.org/10.1038/s41587-020-0439-x). [Online]. Available: <http://www.nature.com/articles/s41587-020-0439-x> (visited on 10/08/2021).
- [178] *Kubernetes/kubernetes*, original-date: 2014-06-06T22:56:04Z, Dec. 17, 2020. [Online]. Available: <https://github.com/kubernetes/kubernetes>.
- [179] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: The condor experience,” *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2, pp. 323–356, Feb. 1, 2005, Publisher: John Wiley & Sons, Ltd, ISSN: 1532-0626. DOI: [10.1002/cpe.938](https://doi.org/10.1002/cpe.938). [Online]. Available: <https://doi.org/10.1002/cpe.938>.
- [180] D. Merkel, “Docker: Lightweight linux containers for consistent development and deployment,” *Linux J.*, vol. 2014, no. 239, Mar. 2014, Place: Houston, TX Publisher: Belltown Media, ISSN: 1075-3583.
- [181] P. Amstutz, M. R. Crusoe, N. Tijani, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Ménager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic, “Common workflow language, v1.0,” 2016. DOI: [10.6084/m9.figshare.3115156.v2](https://doi.org/10.6084/m9.figshare.3115156.v2). [Online]. Available: https://figshare.com/articles/dataset/Common_Workflow_Language_draft_3/3115156.
- [182] “Ensuring accurate resource identification,” *Nature Protocols*, vol. 15, no. 6, pp. 1879–1880, Jun. 1, 2020, ISSN: 1750-2799. DOI: [10.1038/s41596-020-0334-4](https://doi.org/10.1038/s41596-020-0334-4). [Online]. Available: <https://doi.org/10.1038/s41596-020-0334-4>.

- [183] (2020). “Transparency and reproducibility initiative,” SC20, [Online]. Available: <https://sc20.supercomputing.org/submit/transparency-reproducibility-initiative/>.
- [184] R. Iakymchuk, M. B. Vayá, S. Graillat, J. I. Aliaga, and E. S. Quintana-Ortí, “Reproducibility of parallel preconditioned conjugate gradient in hybrid programming environments,” *The International Journal of High Performance Computing Applications*, vol. 34, no. 5, pp. 502–518, 2020. DOI: [10.1177/1094342020932650](https://doi.org/10.1177/1094342020932650). [Online]. Available: <https://doi.org/10.1177/1094342020932650>.
- [185] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, and R. Strackx, “Meltdown: Reading kernel memory from user space,” *Commun. ACM*, vol. 63, no. 6, pp. 46–56, May 2020, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0001-0782. DOI: [10.1145/3357033](https://doi.org/10.1145/3357033). [Online]. Available: <https://doi.org/10.1145/3357033>.
- [186] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, “Spectre attacks: Exploiting speculative execution,” in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1–19. DOI: [10.1109/SP.2019.00002](https://doi.org/10.1109/SP.2019.00002).
- [187] A. K. Sharma and S. Mittal, “Cryptography network security hash function applications, attacks and advances: A review,” in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, 2019, pp. 177–188. DOI: [10.1109/ICISC44355.2019.9036448](https://doi.org/10.1109/ICISC44355.2019.9036448).
- [188] R. C. Merkle, “A digital signature based on a conventional encryption function,” in *Advances in Cryptology CRYPTO 87*, C. Pomerance, Ed., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 1988, pp. 369–378, ISBN: 978-3-540-48184-3. DOI: [10.1007/3-540-48184-2_32](https://doi.org/10.1007/3-540-48184-2_32).
- [189] J. Benet, “IPFS - content addressed, versioned, p2p file system,” *CoRR*, vol. abs/1407.3561, 2014. [Online]. Available: <http://arxiv.org/abs/1407.3561>.
- [190] A. B. Kahn, “Topological sorting of large networks,” *Commun. ACM*, vol. 5, no. 11, pp. 558–562, Nov. 1962, Place: New York, NY, USA Publisher: Association for Computing Machinery, ISSN: 0001-0782. DOI: [10.1145/368996.369025](https://doi.org/10.1145/368996.369025). [Online]. Available: <https://doi.org/10.1145/368996.369025>.
- [191] FITS Working Group. (Aug. 13, 2018). “FITS standard document version 4.0,” [Online]. Available: https://fits.gsfc.nasa.gov/fits_standard.html.
- [192] J. Smith, *Introduction to Digital Filters: With Audio Applications*, ser. Music signal processing series. W3K, 2007, ISBN: 978-0-9745607-1-7. [Online]. Available: <https://books.google.com.au/books?id=pClicQUAsHEC>.
- [193] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd. The MIT Press, 2009, ISBN: 0-262-03384-4.

- [194] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 1, 2020, ISSN: 1476-4687. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). [Online]. Available: <https://doi.org/10.1038/s41586-020-2649-2>.
- [195] M. Frigo and S. Johnson, “The design and implementation of FFTW3,” *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, Feb. 2005, ISSN: 0018-9219. DOI: [10.1109/JPRDC.2004.840301](https://doi.org/10.1109/JPRDC.2004.840301). [Online]. Available: <http://ieeexplore.ieee.org/document/1386650/>.
- [196] L. E. Henry Gomersall, *PyFFTW*, Feb. 2020. [Online]. Available: <https://github.com/pyFFTW/pyFFTW/releases/tag/v0.12.0>.
- [197] P. Vingelmann, F. H. Fitzek, and F. H. Fitzek, *CUDA, release: 10.2.89*. NVIDIA, 2020. [Online]. Available: <https://developer.nvidia.com/cuda-toolkit>.
- [198] L. E. Givon, T. Unterthiner, N. B. Erichson, D. W. Chiang, E. Larson, L. A. Pfister, S. Dieleman, G. R. Lee, S. v. d. Walt, B. Menn, T. M. Moldovan, F. Bastien, X. Shi, J. Schlüter, B. Thomas, C. Capdevila, A. Rubinsteyn, M. M. Forbes, J. Frelinger, T. Klein, B. Merry, N. Merrill, L. Pastewka, L. Y. Liu, S. Clarkson, M. Rader, S. Taylor, A. Bergeron, N. H. Ukani, F. Wang, W.-K. Lee, and Y. Zhou, *Scikit-cuda 0.5.3*, version 0.5.3, May 2019. DOI: [10.5281/zenodo.3229433](https://doi.org/10.5281/zenodo.3229433). [Online]. Available: <https://doi.org/10.5281/zenodo.3229433>.
- [199] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, and A. Fasih, “PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation,” *Parallel Computing*, vol. 38, no. 3, pp. 157–174, 2012, ISSN: 0167-8191. DOI: [10.1016/j.parco.2011.09.001](https://doi.org/10.1016/j.parco.2011.09.001).
- [200] V. Ingle, S. Kogon, and D. Manolakis, *Statistical and Adaptive Signal Processing*. Artech, 2005, ISBN: 978-1-58053-610-3.
- [201] P. Godfrey-Smith, “Induction and confirmation,” in *Theory and Reality An Introduction to the Philosophy of Science*, ser. Science and Its Conceptual Foundations series, Chicago: University of Chicago Press, 2003, pp. 39–56, ISBN: 978-0-226-30061-0.