Journal of Logical and Algebraic Methods in Programming

Algebraic Reasoning for Timeliness-Guided System Design

--Manuscript Draft--

Manuscript Number:	
Article Type:	VSI:WPTE2023-JLAMP- [MGE-Cynthia Kop]
Keywords:	Algebraic Reasoning; Timeliness; System Design; ΔQSD
Corresponding Author:	Neil Davies Predictable Network Solutions Ltd STONEHOUSE, County UNITED KINGDOM
First Author:	Seyed Hossein HAERI
Order of Authors:	Seyed Hossein HAERI
	Peter VAN ROY
	Heinrich APFELMUS
	Peter Thompson
	Neil Davies
	Magne HAVERAAEN
	Mikhail BARASH
	Kevin HAMMOND
	James CHAPMAN
	Artjoms ŠINKAROVS
Manuscript Region of Origin:	UNITED KINGDOM
Abstract:	Designing distributed systems to have predictable performance under all loads is difficult because of resource exhaustion, nonlinearity, and stochastic behaviour. Timeliness, defined as delivering results within a specified delay distribution, is a central aspect of predictable performance. In this paper, we focus on timeliness using the ΔQ Systems Development paradigm (ΔQSD, developed by PNSol), which computes timeliness by modelling systems observationally using so-called outcome expressions. An outcome expression is a compositional definition of a system's observed behaviour that shows the causal connections between its basic operations. Given the ΔQSD operations, we use outcome expressions to compute the stochastic behaviour of the whole system including its timeliness. This paper defines and proves algebraic equivalences of outcome expressions with respect to timeliness. We prove the correctness of known equivalences, and introduce new equivalences and prove their correctness, for outcome expressions containing probabilistic choice, failure, synchronisation (first-to-finish and last-to-finish), and sequential composition operators. We show how to incorporate failure as a first-class citizen in outcome expressions by developing a body of mathematics for improper random variables. The paper shows the practical usefulness of algebraic equivalences by studying the design of a memory system containing a local cache, a networked read with timeout, and the ability to retry. We compute the delay and failure behaviour of several versions of this system, using the equivalences to simplify computations. This work is part of an ongoing project to disseminate and build tool support for ΔQSD, to make it available to the wider community of system designers.

Additional information for Reviewers and Editors

Dear Reviewers and Editors, the submitted article “Algebraic Reasoning for Timeliness-Guided System Design” is a substantially extended and revised version of “Algebraic Reasoning About Timeliness” as presented at ICE 2023.

Namely:
Section 3 (Memory system case study) has been expanded and substantially revised, using more of the algebraic operators so as to give greater illustration of the interlinkage of the formalism and potential practical use.
Section 5 has been heavily modified to become a potential ‘cheat sheet’ for practical application to aid application by ‘practising designers’
New work (section 7) t has identified a re-casting of the work on properisation (section 6) from an abstract algebraic point of view. Appendix A has been added to express the central notion of improper random variables in a measure theory framework

We have revised the introduction and conclusion to incorporate both the additional work and additional context as to the area of applicability of this work as well as adding more figures to help the read to form better intuitions of what is being described.

Highlights

Algebraic Reasoning for Timeliness-Guided System Design

Seyed Hossein HAERI, Peter VAN ROY, Heinrich APFELMUS, Peter W. THOMPSON, Neil J. DAVIES, Magne HAVERAAEN, Mikhail BARASH, Kevin HAMMOND, James CHAPMAN, Artjoms ŠINKAROVS

- Design and analysis of large systems and their performance
- Symbolic reasoning to compute delay and failure behaviour of systems
- Defining systems observationally as basic behaviours plus causality
- Formal proofs of algebraic equivalences for system transformations
- Stochastic approach to predict system delay and failure behaviour
- Extension of ΔQSD paradigm with algebraic reasoning
- Improving efficiency of software tools for system design
- Memory system study with cache, networked read, timeout, and retry
- Modelling failure as a first-class citizen for system design and analysis
- Use of improper random variables to model delay and failure together

Algebraic Reasoning for Timeliness-Guided System Design

Seyed Hossein HAERI[a,b], Peter VAN ROY[d], Heinrich APFELMUS[e], Peter W. THOMPSON[f,c], Neil J. DAVIES[f,c], Magne HAVERAAEN[b], Mikhail BARASH[b], Kevin HAMMOND[g], James CHAPMAN[h], Artjoms ŠINKAROVS[i]

[a] *Input Output Global, Brussels, Belgium*
[b] *Department of Informatics, University of Bergen, Bergen, Norway*
[c] *Predictable Network Solutions Ltd. (PNSol), Stonehouse, UK*
[d] *Université catholique de Louvain, Louvain-la-Neuve, Belgium*
[e] *Cardano Stiftung, Zug, Switzerland*
[f] *Input Output Global, Stonehouse, UK*
[g] *Input Output Global, Fife, UK*
[h] *Input Output Global, Glasgow, UK*
[i] *Input Output Global, Edinburgh, UK*

Abstract

Designing distributed systems to have predictable performance under all loads is difficult because of resource exhaustion, nonlinearity, and stochastic behaviour. Timeliness, defined as delivering results within a specified delay distribution, is a central aspect of predictable performance. In this paper, we focus on timeliness using the ΔQ Systems Development paradigm (ΔQSD, developed by PNSol), which computes timeliness by modelling systems observationally using so-called outcome expressions. An outcome expression is a compositional definition of a system's observed behaviour that shows the causal connections between its basic operations. Given the ΔQSD operations, we use outcome expressions to compute the stochastic behaviour of

Email addresses: hossein.haeri@iohk.io (Seyed Hossein HAERI), pvr@info.ucl.ac.be (Peter VAN ROY), apfelmus@quantentunnel.de (Heinrich APFELMUS), Peter.Thompson@pnsol.com (Peter W. THOMPSON), Magne.Haveraaen@uib.no (Magne HAVERAAEN), Neil.Davies@pnsol.com (Magne HAVERAAEN), mikhail.barash@uib.no (Mikhail BARASH), kevin.hammond@iohk.io (Kevin HAMMOND), james.chapman@iohk.io (James CHAPMAN), artjoms.sinkarovs@iohk.io (Artjoms ŠINKAROVS)

Preprint submitted to Journal of Logical and Algebraic Methods in Programming January 19, 2024

the whole system including its timeliness.

This paper defines and proves algebraic equivalences of outcome expressions with respect to timeliness. We prove the correctness of known equivalences, and introduce new equivalences and prove their correctness, for outcome expressions containing probabilistic choice, failure, synchronisation (first-to-finish and last-to-finish), and sequential composition operators. We show how to incorporate failure as a first-class citizen in outcome expressions by developing a body of mathematics for improper random variables. The paper shows the practical usefulness of algebraic equivalences by studying the design of a memory system containing a local cache, a networked read with timeout, and the ability to retry. We compute the delay and failure behaviour of several versions of this system, using the equivalences to simplify computations. This work is part of an ongoing project to disseminate and build tool support for ΔQSD, to make it available to the wider community of system designers.

Keywords: Algebraic Reasoning, Timeliness, System Design, ΔQSD
PACS: 0000, 1111
2000 MSC: 0000, 1111

---

1. Introduction

Designing distributed systems to have predictable performance under all loads is difficult. At high load, throughput and latency reach their limits and resources such as network capacity, CPU capacity, and storage will be exhausted, causing a dramatic effect on performance. Prediction is difficult because the behaviour of system components and their interactions is both nonlinear and stochastic. For over 20 years, a small group of people associated with the company PNSol has worked on diagnosing and designing systems to predict and correct performance problems [1, 2, 3, 4, 5, 6, 7]. PNSol has developed the ΔQ Systems Development paradigm (ΔQSD) as part of this work. ΔQSD has been used in areas as diverse as telecommunications [8] [9] [10], WiFi [11], and distributed ledgers [12]. ΔQSD has been applied to many large industrial systems, with clients including BT, Vodafone, Boeing Space and Defence, and IOG (formerly IOHK).

This paper defines and proves algebraic properties of the ΔQSD operators with respect to timeliness, i.e., delivering outcomes within the acceptable time-frames. In this paper, our sole resource of concern is time, although

ΔQSD includes other types of resources and their interaction.

This work is part of an ongoing project to disseminate and build tool support for ΔQSD, to make it available to the wider community of system engineers. It is based on on the ΔQSD formalisation given by Haeri et al. [13], which defines outcome expressions and their semantics and gives a real-world example of ΔQSD taken from the blockchain domain.

1.1. Timeliness and the ΔQSD Paradigm

This paper extends the ΔQSD paradigm with algebraic reasoning for timeliness. By facilitating the computation of timeliness, this extension has the potential to greatly improve the efficiency of software tools using ΔQSD, and in addition it allows the use of ΔQSD for back-of-the-envelope estimations by design engineers. We believe that the extension has the potential to have a large positive impact not only on ΔQSD but on system design in general. In this section, we justify this belief by explaining the main concepts of ΔQSD, the systems it targets, and its properties for those systems.

The ΔQSD paradigm models systems using outcome expressions, which are directed graphs that give the causal dependencies between the operation of a system's components. The behaviour of each component is defined by a cumulative distribution function (CDF) of the component's delay (which may depend on various parameters including load). The behaviour of the whole system is given by a CDF that is obtained by combining the CDFs of its components. This may require a significant amount of computation, and this computation must be repeated whenever the system design is modified during the design process. The algebraic equivalences derived in this paper allow simplification of many of these computations.

Scope of ΔQSD. ΔQSD is a systems development paradigm that is able to compute many system properties early on in the design process, such as performance (latency and throughput), timeliness (whether results appear on time), resource consumption, risks, and feasibility. ΔQSD is used for both diagnosis and design:

- *System Diagnosis.* ΔQSD can be used to analyse an existing system, to pinpoint anomalous behaviours so their origin can be found and the system can be corrected.
- *System Design.* ΔQSD can estimate performance trade-offs during the design process. At every step of the design process, performance of

the complete system can be estimated by a computation on the partial design. This computation also determines whether or not the system is feasible, i.e., whether it can or cannot meet the requirements.

Historically, ΔQSD has primarily been used for system diagnosis, to find and solve problems in existing large industrial systems. However, PNSol has recently used ΔQSD for system design, to conceptualize and define the Shelley block diffusion algorithm as used in the Cardano blockchain [13].

Targeted systems. The ΔQSD paradigm primarily targets systems with many independent users where real-time performance is important. This includes systems with large flows of mostly independent data items and systems that are subject to frequent overload situations. Some examples where the paradigm works well:

- Distributed systems that perform tasks for many independent users, such as cryptocurrency platforms.
- Large-scale communications systems including telephony, mobile telephony, content distribution, and publish/subscribe.
- Large client/server systems with many networked connections and back-end databases.
- Distributed sensor networks with real-time data streams and analysis, where edge data is transmitted to cloud tools, analysed, and results return back to the edge.

Note that for systems that execute long sequences of dependent actions, the predictions will be less accurate. We believe it is possible to extend the paradigm for such systems by modelling the dependent sequences, but this has not been the focus of our work.

Properties. The ΔQSD paradigm has the following properties:

- It can define and analyse partially specified systems. System properties can be computed early on in the design process. This is possible even when large parts of the system have not yet been specified concretely, by using approximation that can be refined later. At any point in the design process, the system's latency and throughput can be predicted under various load situations. Infeasible designs can be eliminated early on in the design process.

- It factors the design into three sections: a compositional base made of independent parts; a set of dependencies between parts (e.g., shared resources such as latency, power consumption, storage, and CPU load); and a hierarchy of supervisory systems for risk management.

- It uses a stochastic approach to specify system behaviour, using cumulative distribution functions to model delay and failure. Experience shows that this is a 'sweet spot' that gives good results with respect to the amount of information needed. Predictions are accurate when the system model correctly models both independent and dependent parts.

We note that achieving the full power of the approach can require significant computation (although much less than for simulation-based approaches). Component behaviour is modelled by CDFs, which are functions that potentially contain a large amount of information. Computations done during the design process require convolution, deconvolution, and other arithmetic operations on CDFs. With some attention to efficiency in the formalisation and implementation of ΔQSD, we expect that a modern laptop computer will be sufficient for most practical designs. More information on ΔQSD and many examples of outcome diagrams can be found in the tutorial given at HiPEAC 2023 [14].

1.2. Structure of the Paper and Contributions

This paper is based on a general model theory of resource analysis for systems specified using outcome expressions [15]. That model theory is the first of its kind and we specialise it using the timeliness analysis recipe that is commonly used in ΔQSD (Definition 3). This paper gives a firm mathematical foundation for ΔQSD and uses this to establish important algebraic properties of the ΔQSD operators with respect to timeliness, i.e., when the relevant resource is time. The contributions of this paper are presented in three parts, with two sections for each part:

- Memory system case study (§2 and §3). This part shows the practical application of the results of the paper. It uses these results to study properties of a realistic memory system containing a cache, a networked read with timeout, and the ability to retry when reads time out. This part is of particular interest to system designers, and can be read independently of the rest of the paper.

  - §2 defines the basic concepts of the paper. The most fundamental concept is the *outcome*, which corresponds to an observed system behaviour. Outcomes are combined into outcome expressions with their graphical representation as outcome diagrams. We associate to each outcome a *quality attenuation*, which is also written ΔQ. The quality attenuation is a cumulative distribution function of delay that is allowed to be improper. The use of improper random variables allows modelling failure together with delay. The *timeliness* of an outcome is a relation that compares its observed (or calculated) quality attenuation with its required quality attenuation.
  - §3 gives the case study. We define different versions of a memory system and compute their failure probability and timeliness, using the algebraic equivalences of the paper to simplify the computations. This illustrates the practical usefulness of algebraic equivalences.
  - Fig. 7 in §3 is a table that summarises all the algebraic equivalences of the paper, in a form usable by system designers. The figure is explained in §3.4. Because of its size, the table is split into two parts.

- Algebraic equivalences of outcome expressions with respect to timeliness (§4 and §5). This part gives the main theoretical contributions of the paper.
  - §4 defines the formal syntax and timeliness semantics of outcome expressions. Outcome expressions contain primitive outcomes and four compositional operators: sequential composition; first-to-finish (a.k.a. any-to-finish); last-to-finish (a.k.a. all-to-finish); and probabilistic choice. The graphical representation of an outcome expression is called an outcome diagram. The timeliness semantics associates a cumulative distribution function (CDF) with each outcome expression. As explained before, this CDF is called a *quality attenuation* and it is given the symbol ΔQ. The use of improper distributions is a key property of the ΔQSD approach that allows studying the trade-offs between delay and failure. The example of §3.9, which models timeout and retry, shows the advantage of this approach.

  - §5 derives a series of algebraic equivalences. We first prove a set of known equivalences ("folklore equivalences"). We then prove equivalences for probabilistic choice, distributivity, and failure extraction. For ease of use, these equivalences are summarised in Fig. 7 as mentioned above.

- Algebraic study of failure (§6 and §7). This part gives the theoretical foundations of how failure is incorporated as a first-class citizen in the ΔQSD approach.

  - §6 shows how failure can be extracted from outcomes so that it becomes usable by the algebraic equivalences. This transformation is called *properisation*. Using properisation together with algebraic equivalence allows transformation of an outcome expression to make the failure visible at the top level. This makes it simple to compute the effect of subsystem failure on overall system failure.
  - §7 gives an algebra of failure probabilities and shows how this fits into a more general algebraic model of proper and improper probability distributions. This gives a deeper explanation of the relationship between failure and proper distributions. It shows why the use of improper distributions can be seen as a natural way to combine delay and failure into a single concept.

- Related work (§8). Other formalisms have a much smaller set of transformations than we show here. In a general comparison of ΔQSD with other formalisms, three properties stand out:

  - It gives an observational model of the system that is independent of its internal structure, as opposed to UML, for example, which defines what happens inside the system being modelled.
  - It gives a stochastic definition of system behaviour that covers both delay and failure. This gives good results for the amount of information needed to define the system. In particular, the use of improper random variables makes it easy to explore trade-offs between delay and failure. Classical reliability theory corresponds to a subset of ΔQSD that looks only at failure probabilities.
  - It allows performance and feasibility to be computed for partially defined systems. Determining feasibility is more precise than for

other formalisms because the stochastic approach of ΔQSD allows both independent and dependent outcomes.

§8 gives more information on the relationship between ΔQSD and particular formalisms including BPMN, PEPA, PerformERL, and FMEA.

For brevity, some of the proofs are shortened in this paper. Full proofs can be found in the accompanying technical report [15]. In addition, the technical report shows how the memory system of Fig. 4 can be further elaborated using code running in a Jupyter notebook.

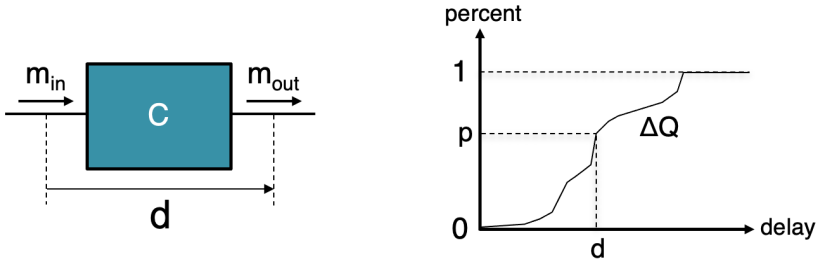
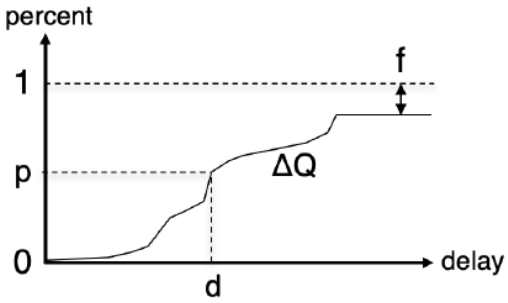
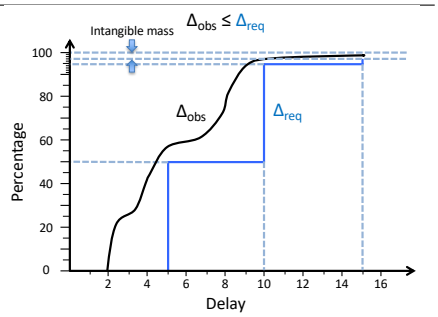


Figure 1: A Component's Operation and its Cumulative Delay Function



(a) Failure is modelled as a quality attenuation whose limit is less than 1.



(b) Timeliness: ΔQ_{obs} (the observed quality attenuation ΔQ) is always to the left and above ΔQ_{req} (the required ΔQ).

Figure 2: Failure and Timeliness

2. Background

We define the basic concepts used in the paper, in terms of systems and their behaviour. These definitions underlie both the practical part (§3) and theoretical part (§4 – §7) of the paper.

Outcome and Quality Attenuation. Consider a component C which inputs message m_{in} and outputs message m_{out} after a delay d. Doing this many times will usually give different delays. We define a cumulative delay function so that p percent of delays are less or equal to d. Fig. 1 gives an illustration.

The ΔQSD paradigm generalises this simple measurement. We measure delay not only for messages, but for all system behaviours that have a starting event and a terminating event. Given a starting event e_{in} and a terminating event e_{out}, what the system gains within the (e_{in}, e_{out}) time frame is called an instance of an *outcome*. We also generalise the property that we measure: we measure not only delay, but any property that makes the system less than perfect. The cumulative distribution function of the property is then called a *quality attenuation* and is denoted by a ΔQ. In what follows, we will consistently use the terms outcome and quality attenuation.

Failure. It is straightforward to generalise quality attenuation to model both delay and failure. It suffices to allow the cumulative delay function's limit to be less than 1. Fig. 2a illustrates this possibility. There is an f percent probability that the delay is infinite, which corresponds precisely to a failure. For the component, it means simply that there is an input message m_{in} with no corresponding output message m_{out} within the timescale of interest. Mathematically, the delay is modelled by a random variable that is allowed to be **improper**: The probability that it is infinite can be greater than 0. This probability is called the intangible mass of the Improper Random Variable (IRV) [16].

The ability to model delay and failure as a single quantity is a key strength of ΔQSD. It makes it easy to explore trade-offs between delay and failure in the system design. This ability shows up clearly in the algebra presented in this paper.

Timeliness. We define *timeliness* as a relation (defined in [8]) between an observed ΔQ_{obs} and a required ΔQ_{req}. We say that the system *satisfies timeliness* for a given outcome if $\Delta Q_{obs} \leq \Delta Q_{req}$. Fig. 2b illustrates this condition. The timeliness relation defines a partial order [13]. If ΔQ_{obs} and

ΔQ_{req} intersect, then the order is undefined. In that case, the probability that the required timeliness is violated is called the *hazard*. It can be computed from the two CDFs.

Outcome Diagram. For a system consisting of multiple interconnected components, one can define a directed graph that combines all the components' outcomes. This graph defines the causal relationships between the outcomes and is called an *outcome diagram*. Each outcome diagram has a corresponding *outcome expression* – a mathematical description of the diagram[1]. Given an outcome expression and the quality attenuations of all its components, it is possible to compute the quality attenuation of the complete system. The reverse process can also be fruitful: given an outcome expression and the required quality attenuation of the complete system, one can estimate the required quality attenuations of its components. This gives the system designer a powerful tool for both design and diagnosis. Many practical examples of outcome diagrams are given in §3 which presents the memory system case study.

Outcome expressions can be manipulated according to algebraic rules, in particular those presented in this paper, which are useful to system designers in ΔQSD. As part of an ongoing project, we are building software tools to support ΔQSD, which can use the algebraic rules presented here for symbolic manipulation of outcome expressions.

3. Memory System Case Study

To show how to use the ΔQSD approach in practice, we study different versions of a memory system consisting of a local cache with a remote main memory. This case study serves four purposes:

- First, it shows how outcome diagrams are used to model nontrivial systems.
- Second, it shows how we can compute properties of the system (failure probability and delay distribution) by algebraic transformations of an outcome diagram. This is often simple enough to be done by hand.

[1]In this paper, we take the equivalence between outcome expressions and outcome diagrams for granted. That equivalence is not the focus of this paper.

- Third, it shows how the full quality attenuation ΔQ of the system can be computed. We combine the distribution functions of the cache's components according to the operations of the outcome diagram. This can be done approximately as a back-of-the-envelope calculation, or more precisely by a software tool that computes with the actual distribution functions.

- Fourth, it shows how to study the trade-off between delay and failure. We extend the memory system to do a retry if a remote read times out. This reduces the probability that the read fails at the price of increased delay.

3.1. Overview of the Case Study

We present the case study in step-by-step fashion.

- We first give the basic definitions used in the examples:
  - We define the memory system by its block diagram (§3.2). This is a standard way to show the system structure.
  - We define the memory system as an outcome diagram (§3.3). This diagram shows the operation of the system, with all messages and the causal connections between them.
  - Fig. 7 gives a table of algebraic equivalences that summarize the results of this paper. These equivalences are used to compute the properties of the examples (§3.4).

- We use the outcome diagram to calculate the failure probability and the delay function. To make this easy to follow, we give three progressively more realistic versions of the cache.
  1. Cache with zero delay network and no timeout (§3.5). We compute the failure probability by transforming the outcome diagram. This shows how the main memory failure affects the probability of overall failure in the presence of the cache.
  2. Cache with reliable network and timeout (§3.6). We compute the cumulative delay distribution. Because of timeout, the failure behavior is more subtle. The cache read always returns a result but it may be a timeout and not a successful read. We show the combined effect of the timeout and the main memory failure.

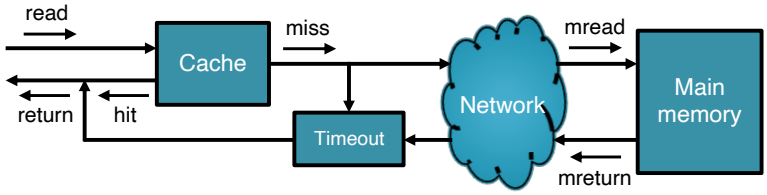


Figure 3: Block Diagram of a Memory System with Local Cache and Networked Main Memory with Timeout

3. Cache with failure-prone network and timeout (§3.7). We show that the overall failure probability of the main memory read across the network is the sum of the probabilities of the network failure and the main memory failure. This confirms that when limited to failure probabilities our approach gives the same result as classical reliability theory.

- We show how to compute the full ΔQ (cumulative delay function) of the memory system (§3.8).
- We show how to extend the memory system with a retry if the networked read times out and we study the trade-off between reduced failure probability of a read versus increased delay (§3.9).

3.2. Block Diagram

Fig. 3 gives the block diagram of the memory system. A read message enters the cache; a cache hit – when the memory word is in the cache – results in an immediate return message; a cache miss – when the memory word is not in the cache – results in a main memory read. The main memory is across a network, so accessing it requires communication in both directions. Main memory access is guarded by a timeout in case of communication failure. The cache miss initialises the timeout timer; the *mreturn* message is passed through if it occurs before the timeout; otherwise, a *timeout* message is passed instead.

3.3. Outcome Diagram

Fig. 4 shows the outcome diagram for the memory system. In this diagram there are six primitive outcomes (the orange circles). These are composed

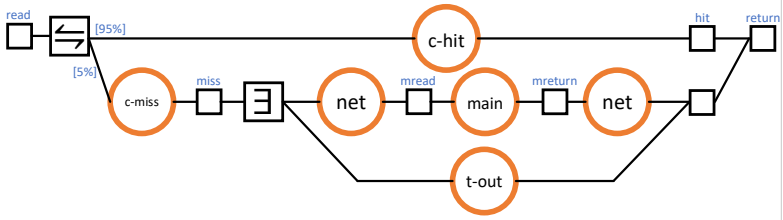


Figure 4: Outcome Diagram for the Memory System of Fig. 3

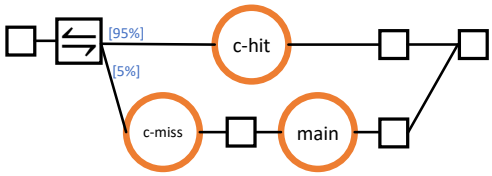


Figure 5: Outcome Diagram for Simplified Cache with Zero Delay Network and No Timeout

with three operations, *sequential composition*, *probabilistic choice*, "⇋", and *first-to-finish* synchronisation, "∃". Probabilistic choice and first-to-finish synchronisation use prefix operators, to make it clear how the component outcomes are combined before they are shown.

We assume that the cache hit rate is 95%. That is modelled using a probabilistic choice with two paths, one for each outcome ("c-hit" and "c-miss"), decorated with their corresponding probabilities. The cache miss leads to a main memory read, which is modelled as a sequential composition with three outcomes, the network send ("net"), the memory read proper ("main"), and the network return ("net"). Timeout is modelled by a first-to-finish relationship between the main memory read and the timer ("t-out").

3.4. Table of Equivalences ("Cheat Sheet")

The rest of §3 studies different versions of the memory system and uses the algebraic equivalences derived in this paper to compute their properties. For ease of consultation, Fig. 7 (in two parts) summarises these equivalences in a "cheat sheet." The equivalences are all written as equations $o_1 = o_2$ where o_1 and o_2 are outcome expressions. Each equivalence states that o_1 and o_2 have strictly the same timeliness, i.e., the quality attenuation is the

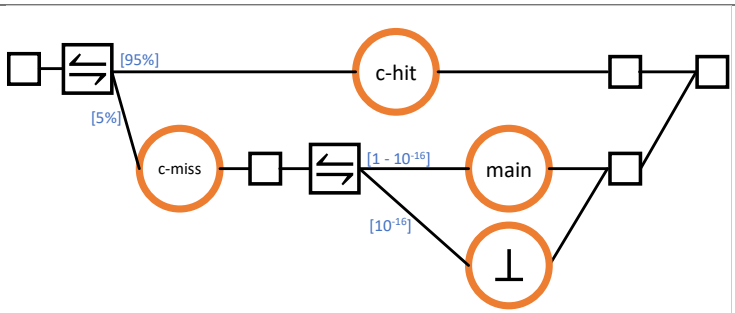


Figure 6: Outcome Diagram for Simplified Cache with Unreliable Main Memory

same for both expressions. In all expressions, $\top$ refers to a system with perfect behaviour (zero delay), basically just a direct connection from input to output, and $\bot$ refers to a failed system (infinite delay), basically just an open connection so that the terminating event never appears.

3.5. Cache with Zero Delay Network and No Timeout

We start with a simplified cache system that has zero network delay and no timeout. Fig. 5 shows the outcome diagram for the simplified cache. This outcome diagram corresponds to the following equivalent outcome expression:

$$\mathit{c\text{-}hit} \overset{95\%}{\rightleftharpoons} (\mathit{c\text{-}miss} \bullet\!\!\rightarrow\!\!\bullet\ \mathit{main}) \tag{1}$$

Let us now assume that the memory read *main* has a small probability of failure, for example 10^{-16}. Using the outcome diagram, we can compute the overall probability of failure when using the cache. We first replace the memory read by a probabilistic choice between a perfectly reliable memory read and a failure $\bot$. Fig. 6 shows the new outcome diagram. The operation of replacing an unreliable operation by a choice between a reliable operation and a failure is called *properisation* (see §6). The new outcome expression is:

$$\mathit{c\text{-}hit} \overset{95\%}{\rightleftharpoons} \left(\mathit{c\text{-}miss} \bullet\!\!\rightarrow\!\!\bullet\ (\mathit{main} \overset{1-10^{-16}}{\rightleftharpoons} \bot)\right) \tag{2}$$

Note that the *main* in Eq. (2) is reliable, unlike the *main* in Eq. (1) which is unreliable. To keep the notation light, this section will always use the same name for both the reliable and unreliable versions, unlike §6 which

$$o_1 \bullet\!\!\rightarrow\!\!\bullet\ (o_2 \bullet\!\!\rightarrow\!\!\bullet\ o_3) = (o_1 \bullet\!\!\rightarrow\!\!\bullet\ o_2) \bullet\!\!\rightarrow\!\!\bullet\ o_3 \qquad o_1 \bullet\!\!\rightarrow\!\!\bullet\ o_2 = o_2 \bullet\!\!\rightarrow\!\!\bullet\ o_1$$
$$\top \bullet\!\!\rightarrow\!\!\bullet\ o = o = o \bullet\!\!\rightarrow\!\!\bullet\ \top \qquad \bot \bullet\!\!\rightarrow\!\!\bullet\ o = \bot = o \bullet\!\!\rightarrow\!\!\bullet\ \bot$$

(a) Equivalences for the Algebraic Structure of $(\mathbb{O}, \bullet\!\!\rightarrow\!\!\bullet)$

$$o_1 \parallel^{\forall} (o_2 \parallel^{\forall} o_3) = (o_1 \parallel^{\forall} o_2) \parallel^{\forall} o_3 \qquad o_1 \parallel^{\forall} o_2 = o_2 \parallel^{\forall} o_1$$
$$\top \parallel^{\forall} o = o = o \parallel^{\forall} \top \qquad \bot \parallel^{\forall} o = \bot = o \parallel^{\forall} \bot$$

(b) Equivalences for Algebraic Structure of $(\mathbb{O}, \parallel^{\forall})$

$$o_1 \parallel^{\exists} (o_2 \parallel^{\exists} o_3) = (o_1 \parallel^{\exists} o_2) \parallel^{\exists} o_3 \qquad o_1 \parallel^{\exists} o_2 = o_2 \parallel^{\exists} o_1$$
$$\bot \parallel^{\exists} o = o = o \parallel^{\exists} \bot \qquad \top \parallel^{\exists} o = \top = o \parallel^{\exists} \top$$

(c) Equivalences for Algebraic Structure of $(\mathbb{O}, \parallel^{\exists})$

Figure 7: The Practising Engineer's Cheat Sheet for Timeliness (Part 1 of 2)

uses $main^{\top}$ for the reliable version. Using an elementary equivalence (from Fig. 7), we move up the failure one level:

$$c\text{-}hit \overset{95\%}{\leftrightharpoons} \left((c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ main) \overset{1-10^{-16}}{\leftrightharpoons} \bot \right) \tag{3}$$

Using an associative equivalence between probabilistic choices (again from Fig. 7), we bring up the failure to the top level:

$$(c\text{-}hit \overset{p'}{\leftrightharpoons} (c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ main)) \overset{q'}{\leftrightharpoons} \bot \tag{4}$$

where the new probabilities are computed as follows:

$$p' = \frac{0.95}{1-(1-0.95)\cdot(1-(1-10^{-16}))} \quad q' = 1-(1-0.95)\cdot(1-(1-10^{-16}))$$

Simplifying gives (in the computation of p', we assume that 10^{-16} can safely be ignored compared to 0.95):

$$(c\text{-}hit \overset{95\%}{\leftrightharpoons} (c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ main)) \overset{1-0.05\cdot 10^{-16}}{\leftrightharpoons} \bot \tag{5}$$

The overall failure rate is therefore $0.05 \cdot 10^{-16}$.

3.6. Cache with Reliable Network and Timeout

Let us now consider the original system of Fig. 4. We assume that the read *main* can fail but the other four primitive outcomes *c-miss*, *c-hit*, *net*, and *t-out* are completely reliable. Fig. 4 has the following outcome expression:

$$c\text{-}hit \overset{95\%}{\leftrightharpoons} (c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ (net \bullet\!\!\rightarrow\!\!\bullet\ main \bullet\!\!\rightarrow\!\!\bullet\ net \parallel^{\exists} t\text{-}out)) \tag{6}$$

$$\bot \leftrightharpoons \bot = \bot \qquad o \bullet\!\!\rightarrow\!\!\bullet \bot = \bot \qquad \top \leftrightharpoons \top = \top \qquad \bot \bullet\!\!\rightarrow\!\!\bullet o = \bot$$
$$\top \bullet\!\!\rightarrow\!\!\bullet o = o \qquad o \bullet\!\!\rightarrow\!\!\bullet \top = o \qquad \top \|^{\forall} o = o \qquad \bot \|^{\exists} o = o$$

$$(o_1 \leftrightharpoons \bot) \bullet\!\!\rightarrow\!\!\bullet o_2 = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons \bot \qquad o_1 \bullet\!\!\rightarrow\!\!\bullet (o_2 \leftrightharpoons \bot) = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons \bot$$
$$(o_1 \leftrightharpoons \top) \bullet\!\!\rightarrow\!\!\bullet o_2 = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons o_2 \qquad o_1 \bullet\!\!\rightarrow\!\!\bullet (o_2 \leftrightharpoons \top) = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons o_1$$

(d) Elementary Equivalences

$$\bot \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} (\bot \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} o) = \bot \underset{\rightharpoonup}{\overset{[p+(1-p)q]}{\leftharpoondown}} o \qquad o_1 \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} \top) = o_2 \underset{\rightharpoonup}{\overset{[q(1-p)]}{\leftharpoondown}} (o_1 \underset{\rightharpoonup}{\overset{\left[\frac{p}{1-q(1-p)}\right]}{\leftharpoondown}} \top)$$

$o_1 \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} o_3) = (o_1 \underset{\rightharpoonup}{\overset{p'}{\leftharpoondown}} o_2) \underset{\rightharpoonup}{\overset{q'}{\leftharpoondown}} o_3$ iff $p' = \frac{p}{1-(1-p)(1-q)}$ and $q' = 1-(1-p)(1-q)$.

$(o_1 \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} o_2) \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} o_3 = o_1 \underset{\rightharpoonup}{\overset{p'}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{q'}{\leftharpoondown}} o_3)$ iff $p' = pq$ and $q' = \frac{q(1-p)}{1-pq}$.

(e) Equivalences with More than One Probabilistic Choice

$$o_1 \underset{\rightharpoonup}{\overset{[p]}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{[q]}{\leftharpoondown}} o_3) = (o_1 \underset{\rightharpoonup}{\overset{[p]}{\leftharpoondown}} o_2) \underset{\rightharpoonup}{\overset{[q]}{\leftharpoondown}} (o_1 \underset{\rightharpoonup}{\overset{[p]}{\leftharpoondown}} o_3) \qquad (o_1 \underset{\rightharpoonup}{\overset{[p]}{\leftharpoondown}} o_2) \underset{\rightharpoonup}{\overset{[q]}{\leftharpoondown}} o_3 = (o_1 \underset{\rightharpoonup}{\overset{[q]}{\leftharpoondown}} o_3) \underset{\rightharpoonup}{\overset{[p]}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{[q]}{\leftharpoondown}} o_3)$$

$$o_1 \|^{\exists} (o_2 \leftrightharpoons o_3) = (o_1 \|^{\exists} o_2) \leftrightharpoons (o_1 \|^{\exists} o_3) \qquad (o_1 \leftrightharpoons o_2) \|^{\exists} o_3 = (o_1 \|^{\exists} o_3) \leftrightharpoons (o_2 \|^{\exists} o_3)$$

$$o_1 \|^{\forall} (o_2 \leftrightharpoons o_3) = (o_1 \|^{\forall} o_2) \leftrightharpoons (o_1 \|^{\forall} o_3) \qquad (o_1 \leftrightharpoons o_2) \|^{\forall} o_3 = (o_1 \|^{\forall} o_3) \leftrightharpoons (o_2 \|^{\forall} o_3)$$

(f) Distributivity

- $(o_1 \underset{\rightharpoonup}{\overset{p_1}{\leftharpoondown}} \bot) \bullet\!\!\rightarrow\!\!\bullet (o_2 \underset{\rightharpoonup}{\overset{p_2}{\leftharpoondown}} \bot) = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \underset{\rightharpoonup}{\overset{[p_1 p_2]}{\leftharpoondown}} \bot$.
- $(o_1 \underset{\rightharpoonup}{\overset{p_1}{\leftharpoondown}} \bot) \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} (o_2 \underset{\rightharpoonup}{\overset{p_2}{\leftharpoondown}} \bot) = (o_1 \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} o_2) \underset{\rightharpoonup}{\overset{r}{\leftharpoondown}} \bot$, where $q = \frac{pp_1}{p_2 - pp_2 + pp_1}$ and $r = p_2 - pp_2 + pp_1$.
- $(o_1 \underset{\rightharpoonup}{\overset{p_1}{\leftharpoondown}} \bot) \|^{\forall} (o_2 \underset{\rightharpoonup}{\overset{p_2}{\leftharpoondown}} \bot) = (o_1 \|^{\forall} o_2) \underset{\rightharpoonup}{\overset{[p_1 p_2]}{\leftharpoondown}} \bot$.
- $(o_1 \underset{\rightharpoonup}{\overset{p_1}{\leftharpoondown}} \bot) \|^{\exists} (o_2 \underset{\rightharpoonup}{\overset{p_2}{\leftharpoondown}} \bot) = \left((o_1 \|^{\exists} o_2) \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} (o_1 \underset{\rightharpoonup}{\overset{q}{\leftharpoondown}} o_2)\right) \underset{\rightharpoonup}{\overset{r}{\leftharpoondown}} \bot$, where $p = \frac{p_1 p_2}{1-(1-p_2)(1-p_1)}$, $q = \frac{p_1(1-p_2)}{p_1+p_2-2p_1p_2}$, and $r = 1-(1-p_2)(1-p_1)$.

(g) Compositional Extraction of Failure

Figure 7: The Practising Engineer's Cheat Sheet for Timeliness (Part 2 of 2)

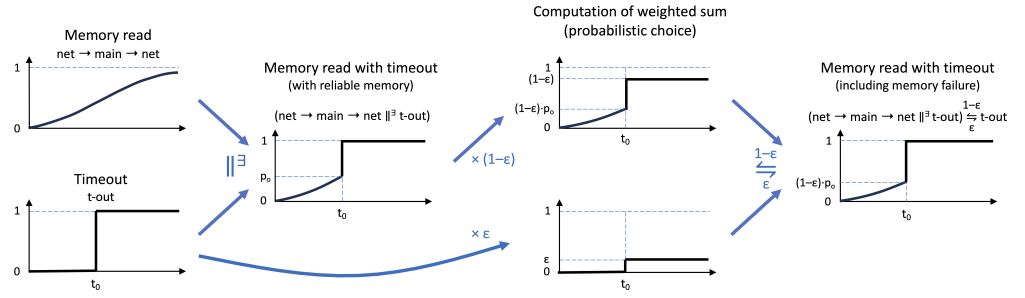


Figure 8: Computing the Delay Function for the Main Memory Read

Doing properisation on the memory read gives:

$$c\text{-}hit \overset{95\%}{\rightleftharpoons} \left(c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet \left(net \bullet\!\!\rightarrow\!\!\bullet \left(main \overset{1-10^{-16}}{\rightleftharpoons} \bot\right) \bullet\!\!\rightarrow\!\!\bullet net \|^{\exists} t\text{-}out\right)\right) \tag{7}$$

We now transform this expression using the equivalences from Fig. 7. The goal is to bring the $\bot$ up to the top level inside a probabilistic choice. We can then read off the reliability directly from this choice. The first step is to bring together the sequential compositions. Using two elementary equivalences we get:

$$c\text{-}hit \overset{95\%}{\rightleftharpoons} \left(c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet \left(\left(net \bullet\!\!\rightarrow\!\!\bullet main \bullet\!\!\rightarrow\!\!\bullet net\right) \overset{1-10^{-16}}{\rightleftharpoons} \bot \|^{\exists} t\text{-}out\right)\right) \tag{8}$$

We focus on the first-to-finish operation:

$$\left(\left(net \bullet\!\!\rightarrow\!\!\bullet main \bullet\!\!\rightarrow\!\!\bullet net\right) \overset{1-10^{-16}}{\rightleftharpoons} \bot \|^{\exists} t\text{-}out\right) \tag{9}$$

We can remove the failure $\bot$ by doing a compositional failure extraction. This gives the following simplified expression:

$$\left(net \bullet\!\!\rightarrow\!\!\bullet main \bullet\!\!\rightarrow\!\!\bullet net \|^{\exists} t\text{-}out\right) \overset{1-10^{-16}}{\rightleftharpoons} t\text{-}out \tag{10}$$

This expression has both a first-to-finish and a probabilistic choice. What kind of cumulative delay does it give? Fig. 8 shows a graphical computation of its cumulative delay function (in our example, $\epsilon = 10^{-16}$). This computation follows the semantics given in §4.2. This figure is read from left to right. Start with the graphs for memory read and timeout on the left. Combining them with a first-to-finish $\|^{\exists}$ gives a memory read with timeout, where the

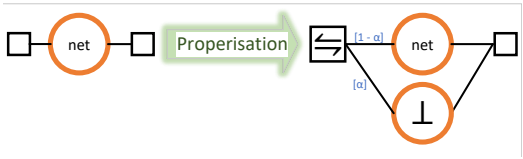


Figure 9: Failure-Prone Network (Properisation of Network Communication)

read is perfectly reliable. On the right we give the analogous delay function for memory read with timeout, where the read can fail. This graph is almost but not quite identical to the previous one. The only difference is that the probability p_0 becomes $p_0 \cdot (1 - \epsilon)$. Given that $\epsilon = 10^{-16}$ is extremely small compared to p_o, we can assume in practice that the graphs are identical. We can therefore simplify Eq. (10) into a pure timeout:

$$(net \bullet\!\!\rightarrow\!\!\bullet\ main \bullet\!\!\rightarrow\!\!\bullet\ net \|^{\exists} t\text{-}out) \tag{11}$$

We replace this in Eq. (8), which gives:

$$c\text{-}hit \overset{95\%}{\rightleftharpoons} (c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ (net \bullet\!\!\rightarrow\!\!\bullet\ main \bullet\!\!\rightarrow\!\!\bullet\ net \|^{\exists} t\text{-}out)) \tag{12}$$

The main memory failure is therefore hidden by the timeout.

3.7. Cache with Failure-Prone Network and Timeout

Assuming that the network is completely reliable is unrealistic. Let us now assume that both the network *net* and the read *main* can fail. On the other hand, we assume that the cache and timeout are completely reliable. This is realistic since both are implemented locally by robust circuitry. Fig. 9 shows a failure-prone network communication. We can replace the two network communications in the remote memory access by failure-prone networks. Fig. 10 gives the outcome diagram. This can be written as an outcome expression:

$$(net \overset{1-\alpha}{\rightleftharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet\ (main \overset{1-\epsilon}{\rightleftharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet\ (net \overset{1-\alpha}{\rightleftharpoons} \bot) \tag{13}$$

Each network communication has a failure probability of α. A typical value of α is around 10^{-10} for wired local area networks such as Ethernet. The main memory read has a typical failure probability of 10^{-16}. Doing two applications of failure extraction, we transform this into:

$$(net \bullet\!\!\rightarrow\!\!\bullet\ main \bullet\!\!\rightarrow\!\!\bullet\ net) \overset{(1-\epsilon)\cdot(1-\alpha)^2}{\rightleftharpoons} \bot \tag{14}$$

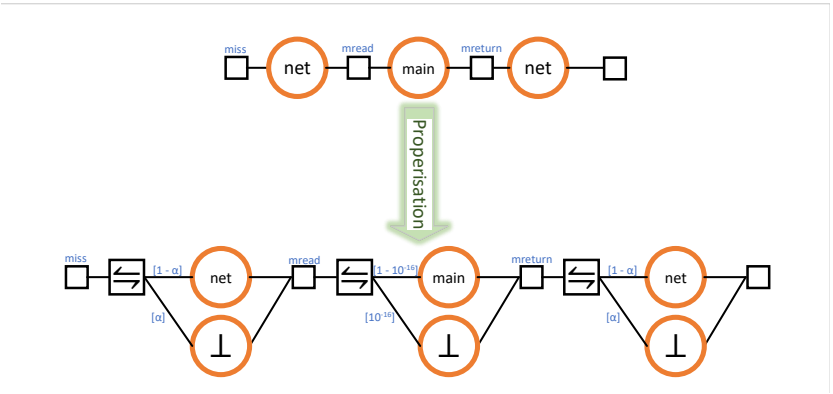


Figure 10: Remote Memory Access when the Network can Fail

The failure rate is $1 - (1 - \epsilon) \cdot (1 - \alpha)^2$. If both α and ϵ are small, this failure rate is closely approximated by $\epsilon + 2\alpha$, which is $2 \cdot 10^{-10}$ (in this case, we ignore ϵ compared to α). This is the same result as given by classical reliability theory.

3.8. Direct Computation of Quality Attenuation ΔQ

We now show how to compute the full quality attenuation ΔQ of the cache example using the semantics given in this paper. This is a numeric computation using the CDF values of the ΔQs. In general, this computation cannot easily be done by hand, but it is straightforward with a software tool. It gives much more information than just a failure probability. As an illustration, Fig. 8 gives a visual example of what such a CDF value would look like. First define a new outcome *mem*:

$$mem = net \bullet\!\!\rightarrow\!\!\bullet\ (main \overset{1-10^{-16}}{\rightleftharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet\ net \tag{15}$$

which corresponds to a networked main memory read. We start by computing its quality attenuation $\Delta\mathrm{Q}_{mem}$:

$$\Delta\mathrm{Q}_{mem} = \Delta\mathrm{Q}_{net} * ((1 - 10^{-16}) \times \Delta\mathrm{Q}_{main} + 10^{-16} \times \Delta\mathrm{Q}_{\bot}) * \Delta\mathrm{Q}_{net} \tag{16}$$

Since $\Delta\mathrm{Q}_{\bot} = \mathbf{0}$, we can simplify this to:

$$\Delta\mathrm{Q}_{mem} = \Delta\mathrm{Q}_{net} * (1 - 10^{-16}) \times \Delta\mathrm{Q}_{main} * \Delta\mathrm{Q}_{net} \tag{17}$$

The overall ΔQ_{obs} is then given by:

$$\Delta Q_{obs} = 0.95 \times \Delta Q_{c\text{-}hit} + 0.05 \times (\Delta Q_{c\text{-}miss} * (\Delta Q_{mem} + \Delta Q_{t\text{-}out} - \Delta Q_{mem} \times \Delta Q_{t\text{-}out})). \tag{18}$$

In these expressions $\times$ is arithmetic multiplication and $*$ is convolution. This computation gives us the CDF for the observed execution time of a memory read. For readers interested in seeing fully worked-out numerical examples, we recommend looking up the tutorial [14].

Given the observed timeliness ΔQ_{obs} as computed above, we can check whether it satisfies the requirement. §2 defines this check as $\Delta Q_{obs} \leq \Delta Q_{req}$ where ΔQ_{req} is the required timeliness. As explained in §2, this relation is a partial order between two CDFs. If the CDFs intersect, then there is a nonzero probability that the timeliness is violated.

3.9. Cache with Timeout and Retry

The ΔQ concept combines both delay and failure in a single value. This is a powerful property that can help us when designing a system. It lets us easily design systems that trade off delay for failure or vice versa. In this section we show it for our memory system.

If the network is slow then networked reads may time out too often. There are two possible fixes at the client side: either increase the timeout delay or do a retry. Both fixes reduce failure rate at the cost of increasing delay. If the timeout is due to lost packets, then increasing the timeout delay will have no effect. In that case it is better to do a retry. We model the retry by an outcome expression and we compute the effect of the retry on failure rate by using algebraic equivalences. In what follows, we abbreviate the networked read ($net \bullet\!\!\rightarrow\!\!\bullet\ main \bullet\!\!\rightarrow\!\!\bullet\ net$) as *netmain*.

Our goal is to compute the probability of a successful read with up to one retry. To be precise, if a networked read attempt times out, then we do another networked read attempt. The overall read operation will only fail if the second attempt times out. Recall that in Eq. (11), we gave an outcome expression for a networked read with timeout, which we can write as follows:

$$(netmain \parallel^{\exists} t\text{-}out) \tag{19}$$

This expression gives the outcome for a successful return signal. This should not be confused with a successful read. The memory system will *always* eventually give a successful return signal, simply because when the read fails

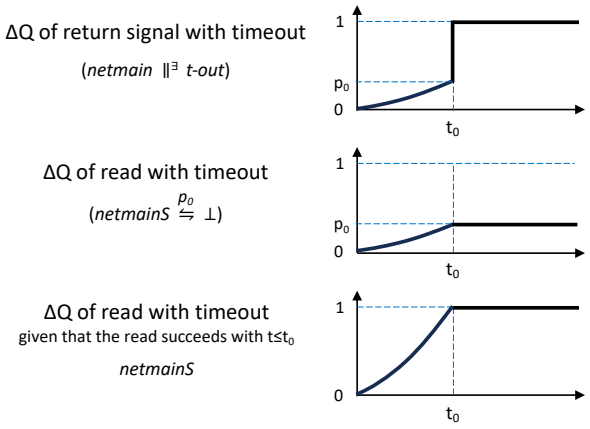


Figure 11: Cumulative Delay for Return Signal versus Read

the return signal comes from the timeout. For a successful read with timeout, the outcome expression is:

$$(netmainS \overset{p_0}{\leftrightarrows} \bot) \tag{20}$$

where $netmainS$ is a networked read given that the read succeeds with delay $t \leq t_0$. Fig. 11 compares the cumulative delay functions. We model the timeout as a probabilistic choice between success ($netmainS$ with probability p_0) and failure ($\bot$ with probability $1 - p_0$). Here the probability p_0 corresponds to a successful read with $t \leq t_0$, which is $p_0 = \Delta\mathrm{Q}_{netmain}(t_0)$ (see Fig. 8). We model the retry by extending Eq. (20):

$$netmainS \overset{p_0}{\leftrightarrows} (t\text{-}out \bullet\!\!\rightarrow\!\!\bullet (netmainS \overset{p_0}{\leftrightarrows} \bot)) \tag{21}$$

We have replaced the failure $\bot$ in Eq. (20) by a timeout followed by another networked read. To compute the failure probability of this expression, we use the equivalences of Fig. 7 to transform this into:

$$(netmainS \overset{p'}{\leftrightarrows} (t\text{-}out \bullet\!\!\rightarrow\!\!\bullet netmainS)) \overset{q'}{\leftrightarrows} \bot \tag{22}$$

The failure is now visible at the top level. The probability of success is q'. From Fig. 7 the probabilities p' and q' are computed as follows:

$$p' = \frac{p_0}{1-(1-p_0)^2} \qquad q' = 1 - (1 - p_0)^2$$

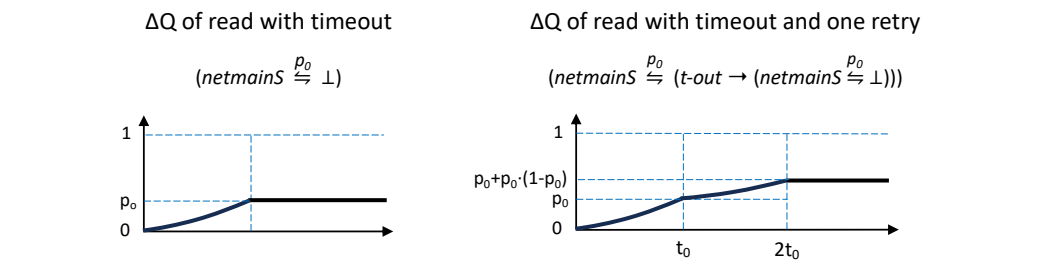


Figure 12: Cumulative Delay for Read with No Retry and One Retry

This gives $q' = 2p_0 - {p_0}^2 = (2 - p_0)p_0$. Recall that with no retry the success probability was p_0 (from Eq. (20)). With one retry the success probability has therefore increased from p_0 to $(2 - p_0)p_0$. Fig. 12 compares the ΔQs of the two cases. This approach can easily be extended to any number of retries. The success probability can be computed from the number of retries, the timeout delay t_0, and $\Delta Q_{netmain}$.

3.10. Closing Remarks on the Case Study

This section has shown how the use of algebraic equivalences can simplify computations of failure and timeliness on realistic systems. The memory system of this section is simplified for pedagogical purposes, but it contains the elements of more complex systems. More realistic memory systems give rise to more complicated outcome diagrams in which failure (⊥) appears at multiple levels. The techniques of this section can be used to combine these failures to compute the overall failure probability and the effect of failure on the memory system. Furthermore, the techniques allow comparison of designs with different trade-offs between delay and failure.

The probability of main memory failure (10^{-16}) may seem small, but it is wrong to ignore such small numbers. They can increase by being combined with probabilities from other parts of the system and by being part of operations repeated many times (for instance a typical processor may perform 10^9 memory accesses per second). The ΔQSD approach keeps track of them and allows the designer to decide when and where they can be ignored.

4. Syntax and Timeliness Semantics

In this section, we present the formal syntax and semantics of outcome expressions (Definition 1), as given in our earlier work [13]. One can give

multiple semantices to that syntax, depending on the resource of interest. Our focus here is on time, so we present a timeliness semantics for outcome expressions (Definition 3) from our earlier work [13]. That syntax and semantics give us a language together for communicating the algebraic properties of outcome expressions w.r.t. time, as outlined in §4.3, which will be used in the remainder of this paper. The keen reader is invited to consult our accompanying technical report [15] for the additional details.

4.1. Syntax of Outcome Expressions

Definition 1 (Haeri et al. [13])**.** Assume a set $\overline{\mathbb{B}}$ of primitive outcomes. We use variables $\beta \in \overline{\mathbb{B}}$ to represent individual primitive outcomes. We define the abstract syntax of outcome expressions as follows:

$$
\begin{array}{rcll}
\mathbb{O} \ni o & ::= & \beta & \text{primitive outcome} \\
& | & o \bullet\!\!\rightarrow\!\!\bullet\, o' & \text{sequential composition} \\
& | & (o \parallel^{\forall} o') & \text{all-to-finish (a.k.a. last-to-finish)} \\
& | & o \overset{p}{\rightleftharpoons} o' & \text{probabilistic choice} \\
& | & (o \parallel^{\exists} o') & \text{any-to-finish (a.k.a. first-to-finish).}
\end{array}
$$

This defines outcome expressions as combinations of primitive outcomes β and four composition operators. In the case of probabilistic choice, the numerical value p ($0 \leq p \leq 1$) is the probability of choosing the left alternative; correspondingly, $(1 - p)$ is the probability of choosing the right alternative. We distinguish two constant outcomes: $\top$ for "perfection" and $\bot$ for "unconditional failure."

Note that the operator "$\exists$" in the outcome diagrams is "$\parallel^{\exists}$" in the outcome expressions. That is to signify that when two outcomes are connected by any-to-finish, they are performed concurrently; hence the "$\parallel$" sign. One need not emphasise that concurrency in the outcome diagrams because our left-to-right directional convention on causal dependency already implies concurrency when forking off an "$\exists$" in the outcome diagrams. Similarly, for "$\parallel^{\forall}$" in the outcome expressions, the sign in the outcome diagrams is simply "$\forall$".

4.2. Timeliness Semantics for Outcome Expressions

Having defined the syntax of outcome expressions, we now want to give them a semantics in terms of timeliness. This semantics maps each outcome expression $o \in \mathbb{O}$ to an improper probability distribution $\Delta\mathrm{Q}[\![o]\!]$ that represents the quality attenuation of this outcome, i.e. the distribution of its time delay.

We model the time delay of an outcome by a *cumulative distribution function* (CDF) $\Delta Q(t)$, which records the probability that the outcome occurs in a time $t_0 \leq t$. Such a function is monotonically increasing and takes values between $0 = 0\%$ and $1 = 100\%$ inclusive. In addition, we allow for the possibility that the outcome never occurs at all in a finite amount of time — the outcome may fail, and this manifests in the fact that the limit $\lim_{t\to\infty} \Delta Q(t)$ may stay strictly below 1. A cumulative distribution function of this kind describes an *improper probability distribution* [16]. The *intangible mass* $\Im(\Delta\mathrm{Q}) = 1 - \lim_{t\to\infty} \Delta Q(t)$ of such a distribution, also known as the *failure probability*, is the probability that the outcome never occurs. Notably, we make no distinction between an infinite delay and an early failure due to an exception, as these two kinds of outcomes tend to be indistinguishable in a practical setting.

We use the letter $\mathbb{I}$ to denote the set of these improper probability distributions. For practical computations, the cumulative distribution function $Q(t)$ is a good way to describe an element $Q \in \mathbb{I}$. However, this element can also be viewed as a probability measure on the set of time delays. For a continuous distribution of time delays, the function $Q(t)$ is differentiable, and its derivative $Q'(t)$ is the *probability density function* (PDF) of the distribution. However, we also allow discrete time delays, which are described by point measures. We refer to Appendix A for the precise definition of improper probability distributions $\mathbb{I}$, where we also define the convolution $*$ of two improper probability distributions.

Fix a countable set of ΔQ variables Δ_v. We define $\Delta = \Delta_v \cup \mathbb{I}$ to denote both IRVs and ΔQ variables.

We first define a mapping between primitive outcomes $\overline{\mathbb{B}}$ and ΔQs.

Definition 2. We call a function $\Delta_\circ[\![.]\!] : \overline{\mathbb{B}} \to \Delta$ a *basic assignment* when $\Delta_\circ[\![\top]\!] = \mathbf{1}$ and $\Delta_\circ[\![\bot]\!] = \mathbf{0}$. Here, $\mathbf{1}$ is the distribution that succeeds immediately, with cumulative distribution function $\mathbf{1}(t) := 1$ for all t. Conversely, $\mathbf{0}$ is the distribution that always fails, $\mathbf{0}(t) := 0$ for all t.

We now define the semantics of an outcome expression as a mapping between the outcome expression and an IRV, for a given basic assignment.

Definition 3 (Haeri et al. [13])**.** Given a basic assignment $\Delta_\circ[\![.]\!] : \overline{\mathbb{B}} \to \Delta$,

define $\Delta\mathrm{Q}[\![.]\!]_{\Delta_\circ} : \mathbb{O} \to \mathbb{I}$ such that

$$
\begin{array}{ll}
\Delta\mathrm{Q}[\![\beta]\!]_{\Delta_\circ} & = \begin{cases} \mathbf{1} & \text{when } \Delta_\circ[\![\beta]\!] \notin \mathbb{I} \\ \Delta_\circ[\![\beta]\!] & \text{otherwise} \end{cases} \\
\Delta\mathrm{Q}[\![o \bullet\!\!\rightarrow\!\!\bullet\, o']\!]_{\Delta_\circ} & = \Delta\mathrm{Q}[\![o]\!]_{\Delta_\circ} * \Delta\mathrm{Q}[\![o']\!]_{\Delta_\circ} \\
\Delta\mathrm{Q}[\![o \overset{p}{\rightleftharpoons} o']\!]_{\Delta_\circ} & = p \times \Delta\mathrm{Q}[\![o]\!]_{\Delta_\circ} + (1-p) \times \Delta\mathrm{Q}[\![o']\!]_{\Delta_\circ} \\
\Delta\mathrm{Q}[\![o \parallel^{\forall} o']\!]_{\Delta_\circ} & = \Delta\mathrm{Q}[\![o]\!]_{\Delta_\circ} \times \Delta\mathrm{Q}[\![o']\!]_{\Delta_\circ} \\
\Delta\mathrm{Q}[\![o \parallel^{\exists} o']\!]_{\Delta_\circ} & = \Delta\mathrm{Q}[\![o]\!]_{\Delta_\circ} + \Delta\mathrm{Q}[\![o']\!]_{\Delta_\circ} - \Delta\mathrm{Q}[\![o]\!]_{\Delta_\circ} \times \Delta\mathrm{Q}[\![o']\!]_{\Delta_\circ}
\end{array}
$$

Here, the notation $*$ denotes the convolution of two improper probability distributions, and the operations $+, -$, and $\times$ operate on the cumulative distribution functions.

In what follows, we will drop $\Delta_\circ$ whenever the basic assignment is fixed throughout a computation.

One way to interpret Definition 3 is that $\Delta\mathrm{Q}[\![.]\!]_{\Delta_\circ}$ is a homomorphism from the term algebra of outcome expressions $\mathbb{O}$ to an algebra of probability distributions $\mathbb{I}$, as discussed in §7.

Remark 1. Note that, according to Definition 3, we get $\Delta\mathrm{Q}[\![o_1 \bullet\!\!\rightarrow\!\!\bullet\, o_2]\!] = \Delta\mathrm{Q}[\![o_2 \bullet\!\!\rightarrow\!\!\bullet o_1]\!]$. This may seem counter-intuitive because $o_1 \bullet\!\!\rightarrow\!\!\bullet o_2 \neq o_2 \bullet\!\!\rightarrow\!\!\bullet o_1$. $\Delta\mathrm{Q}[\![o_1 \bullet\!\!\rightarrow\!\!\bullet\, o_2]\!] = \Delta\mathrm{Q}[\![o_2 \bullet\!\!\rightarrow\!\!\bullet\, o_1]\!]$ is, nonetheless, valid because, intuitively, $o_1 \bullet\!\!\rightarrow\!\!\bullet\, o_2$ is just as timely as $o_2 \bullet\!\!\rightarrow\!\!\bullet\, o_1$. See the proof of Theorem 2 in our accompanying technical report [15] for the mathematical justification of that intuition. □

4.3. Connecting Algebra to Timeliness

An algebraic structure typically consists of a carrier set, a few operations on the carrier set, and a finite set of identities that those operations need to satisfy.

- Given our intentions for the generality of the algebraic study of resource consumption à la ΔQSD, the natural carrier set will be $\mathbb{O}$.[2]
- The full set of operators on $\mathbb{O}$ is $\mathcal{P} = \{\bullet\!\!\rightarrow\!\!\bullet, \parallel^{\forall}, \parallel^{\exists}, \rightleftharpoons\}$. However, most algebraic structures do not need all those operators. Different structures work with different number of operations; for example, a monoid works with only one operation; whilst a group works with two.

[2]When more appropriate, one may choose to take more specific carrier sets. See §7 for an example.

- Finally, the identities are of the form $o_l = o_r$.

We take $\Delta\mathrm{Q}[\![.]\!]$ (Definition 3) to be the model of time consumption for $\mathbb{O}$. We write $\odot\odot\, time \vDash o_l = o_r$ when $\Delta\mathrm{Q}[\![o_l]\!] = \Delta\mathrm{Q}[\![o_r]\!]$. That is when o_l and o_r are as timely. For example, unsurprisingly, $o_1 \parallel^{\exists} o_2 = o_2 \parallel^{\exists} o_1$ because $\Delta\mathrm{Q}[\![o_1 \parallel^{\exists} o_2]\!] = \Delta\mathrm{Q}[\![o_2 \parallel^{\exists} o_1]\!]$.

With time being our solo resource of interest in this paper, we tend to drop the initial "$\odot\odot\, time \vDash$" from the above formulation hereafter.

5. Algebraic Results

This section is a summary of the algebraic results obtained using the machinery developed in the previous section. §5.1 gives the algebraic structures the ΔQSD operators ($\mathcal{P}$) form with the set of outcome expressions ($\mathbb{O}$). It also explains why those operators fail to establish stronger algebraic structures. §5.2 utilises the footing given by Definition 3 to prove a series of ΔQSD equivalences that have been in use by the ΔQSD practitioners in large-scale real-world projects. Next, §5.3 focuses on distributivity of the ΔQSD operators ($\mathcal{P}$). Finally, §5.4 is on compositionally extracting failure. This is important for systematically *pushing* the failure rate up to the top of the outcome expressions so one can swiftly assess the failure rate of a given outcome expression (and, hence, a design) without going through all the details of Definition 3. To that end, Theorem 7 gives instructions for each an every ΔQSD operator.

5.1. Algebraic Structures

This section establishes several important properties on $\mathbb{O}$:

- probabilistic choice forms a magma (Theorem 1);
- sequential composition forms a commutative monoid with $\top$ and $\bot$ as the identity and absorbing elements (Theorem 2);
- all-to-finish forms a commutative monoid with $\top$ and $\bot$ as the identity and absorbing elements (Theorem 3);
- any-to-finish forms a commutative monoid with $\bot$ and $\top$ as the identity and absorbing elements (Theorem 4); and

- neither all-to-finish nor any-to-finish nor their combination form the familiar richer algebraic structures (Remarks 2, 3, and 4).

Theorem 1. $(\mathbb{O}, \leftrightharpoons)$ *forms a magma when observing time.*

A magma is the weakest algebraic structure. We cannot derive anything stronger because $\leftrightharpoons$ is not even associative; although expressions containing two consecutive occurrences of $\leftrightharpoons$ can be re-associated, but the coefficients will change. Lemmas 2 and 3 give the exact formulae.

Theorem 2. *When observing time,* $(\mathbb{O}, \bullet\!\!\rightarrow\!\!\bullet)$ *forms a commutative monoid with* $\top$ *and* $\bot$ *as the identity and absorbing elements, respectively.*

Theorem 3. *When observing time,* $(\mathbb{O}, \|^{\forall})$ *forms a commutative monoid with* $\top$ *and* $\bot$ *as the identity and absorbing elements, respectively.*

Remark 2. It is important to notice that, when observing time, $(\mathbb{O}, \|^{\forall})$ does *not* form a group because an outcome has no inverse element; intuitively, one can never undo an outcome!

In order to prove that claim formally, suppose otherwise. That is, suppose that there exist a pair of outcomes o_1 and o_2 such that $o_1 \|^{\forall} o_2 = \top$. Then, $\Delta\mathrm{Q}[\![o_1 \|^{\forall} o_2]\!] = \Delta\mathrm{Q}[\![\top]\!]$ which implies $\delta_1 \times \delta_2 = \mathbf{1} \Rightarrow \delta_2 = \frac{\mathbf{1}}{\delta_1}$. However, given that $\delta_1 \leq \mathbf{1}$, we get $\delta_2 \geq \mathbf{1}$. The latter inequality can only be satisfied when $o_1 = \top$. Restricting the application of ΔQSD to perfection is not practical. □

Theorem 4. *When observing time,* $(\mathbb{O}, \|^{\exists})$ *forms a commutative monoid with* $\bot$ *and* $\top$ *as the identity and absorbing elements, respectively.*

Remark 3. Similar to the case for $\|^{\forall}$, it is important to note that, when observing time, $(\mathbb{O}, \|^{\exists})$ does not form a group. Again, it is the lack of an inverse element that is causing the trouble. Following our previous result, suppose that there exist a pair of outcomes o_1 and o_2 such that $o_1 \|^{\exists} o_2 = \bot$. Then, $\Delta\mathrm{Q}[\![o_1 \|^{\exists} o_2]\!] = \Delta\mathrm{Q}[\![\bot]\!]$ which implies $\delta_1 + \delta_2 - \delta_1 \times \delta_2 = \mathbf{0} \Rightarrow \delta_2 = \frac{\delta_1}{\delta_1 - 1}$. However, because $\delta_1 \leq \mathbf{1}$, we get $\delta_2 \leq \mathbf{0}$. But, only $\bot$ can satisfy the latter inequality. There is no reason to develop a system in which all the outcomes will fail unconditionally! □

Having established that both $(\mathbb{O}, \|^{\forall})$ and $(\mathbb{O}, \|^{\exists})$ form commutative monoids for time, a natural question is whether $(\mathbb{O}, \|^{\forall}, \|^{\exists})$ or $(\mathbb{O}, \|^{\exists}, \|^{\forall})$ form semirings. This is not the case, since they do not distribute over one another.

Lemma 1 helps Remark 4 show how the desirable distributivities fail.

$$\bot \leftrightharpoons \bot = \bot \qquad o \bullet\!\!\rightarrow\!\!\bullet \bot = \bot \qquad \top \leftrightharpoons \top = \top \qquad \bot \bullet\!\!\rightarrow\!\!\bullet o = \bot$$
$$\top \bullet\!\!\rightarrow\!\!\bullet o = o \qquad o \bullet\!\!\rightarrow\!\!\bullet \top = o \qquad \top \|^{\forall} o = o \qquad \bot \|^{\exists} o = o$$

$$(o_1 \leftrightharpoons \bot) \bullet\!\!\rightarrow\!\!\bullet o_2 = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons \bot \qquad o_1 \bullet\!\!\rightarrow\!\!\bullet (o_2 \leftrightharpoons \bot) = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons \bot$$
$$(o_1 \leftrightharpoons \top) \bullet\!\!\rightarrow\!\!\bullet o_2 = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons o_2 \qquad o_1 \bullet\!\!\rightarrow\!\!\bullet (o_2 \leftrightharpoons \top) = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \leftrightharpoons o_1$$

$$\bot \overset{p}{\leftrightharpoons} (\bot \overset{q}{\leftrightharpoons} o) = \bot \overset{[p+(1-p)q]}{\leftrightharpoons} o \qquad o_1 \overset{p}{\leftrightharpoons} (o_2 \overset{q}{\leftrightharpoons} \top) = o_2 \overset{[q(1-p)]}{\leftrightharpoons} (o_1 \overset{\left[\frac{p}{1-q(1-p)}\right]}{\leftrightharpoons} \top)$$

Figure 13: Folklore Equivalences Containing $\top$ and $\bot$

Lemma 1. $\odot\odot$ *time* $\vDash o_1 \|^{\exists} o_2 = \top$ *implies* $o_1 = \top$ *and* $o_2 = \top$.

Remark 4. Neither $(\mathbb{O}, \|^{\forall}, \|^{\exists})$ nor $(\mathbb{O}, \|^{\exists}, \|^{\forall})$ form a semi-ring when observing time: for this to be the case, $\|^{\forall}$ and $\|^{\exists}$ would need to distribute over one another. The first distributivity requirement is:

$$o_1 \|^{\exists} (o_2 \|^{\forall} o_3) \overset{?}{=} (o_1 \|^{\exists} o_2) \|^{\forall} (o_1 \|^{\exists} o_3) \tag{23}$$

Equating $\Delta\mathrm{Q}[\![.]\!]$s of the two sides, one eventually makes it to the requirement that either $\delta_1 = \mathbf{0}$ or $\Delta\mathrm{Q}[\![(o_1 \|^{\exists} o_3) \|^{\exists} o_2]\!] = \top$. In other words, it follows by Lemma 1 that Eq. (23) can only hold under the trivial conditions when either $o_1 = \bot$ or $o_1 = o_2 = o_3 = \top$. The second distributivity requirement is

$$o_1 \|^{\forall} (o_2 \|^{\exists} o_3) \overset{?}{=} (o_1 \|^{\forall} o_2) \|^{\exists} (o_1 \|^{\forall} o_3) \tag{24}$$

Again, equating $\Delta\mathrm{Q}[\![.]\!]$s of the two sides, one eventually comes to observe that Eq. (24) only holds when $\delta_1 = \mathbf{1} \wedge \delta_2 \neq \mathbf{0} \wedge \delta_3 \neq \mathbf{0}$, i.e., when $o_1 = \top \wedge o_2 \neq \bot \wedge o_3 \neq \bot$. □

5.2. Folklore Equivalences Containing Constant Outcomes

Practitioners of ΔQSD already use some algebraic equivalences to simplify outcome expressions for timeliness analysis. These "folklore equivalences," shown in Fig. 13, all contain constant outcomes ($\top$ or $\bot$). These equivalences provide the basis for rewrite rules that are useful for construction of normal forms, such as expressing a given system as a convolution of probabilistic choices or a probabilistic choice of convolutions. Such rewriting allows for: extraction of common sub-expressions permitting aggregation of

failure rates (distinguishing between conditional and unconditional failure); identifying minimal delays; highlighting branching probabilities to identify issues of relative criticality; and more. Identification of minimal delays is useful for quickly assessing whether a particular outcome diagram is *feasible* without having to compute the complete ΔQ (see §3, for example). In addition, the equivalences of Fig. 13 are very handy in the proofs of properties such as those established in this paper. Two examples are the proofs of Theorem 7 of [17] and Theorem 7 here.

Before we delve into Fig. 13, we prove a result about re-associating probabilistic choice. Given an expression with two consecutive probabilistic choices, one of which wrapped inside a pair of parentheses, the ΔQSD practitioner might be interested in wrapping the other two inside a pair of parentheses – re-associating the probabilistic choices, in effect. Lemmata 2 and 3 give the conditions on the coefficients of those probabilistic choices.

Lemma 2. $o_1 \overset{p}{\leftharpoondown\!\!\rightharpoonup} (o_2 \overset{q}{\leftharpoondown\!\!\rightharpoonup} o_3) = (o_1 \overset{p'}{\leftharpoondown\!\!\rightharpoonup} o_2) \overset{q'}{\leftharpoondown\!\!\rightharpoonup} o_3$ *iff* $p' = \frac{p}{1-(1-p)(1-q)}$ *and* $q' = 1-(1-p)(1-q)$.

Lemma 3. $(o_1 \overset{p}{\leftharpoondown\!\!\rightharpoonup} o_2) \overset{q}{\leftharpoondown\!\!\rightharpoonup} o_3 = o_1 \overset{p'}{\leftharpoondown\!\!\rightharpoonup} (o_2 \overset{q'}{\leftharpoondown\!\!\rightharpoonup} o_3)$ *iff* $p' = pq$ *and* $q' = \frac{q(1-p)}{1-pq}$.

Theorem 5. *The equivalences in Fig. 13 are correct.*

Proof. We will only present the proof of $\bot \overset{p}{\leftharpoondown\!\!\rightharpoonup} \bot = \bot$ here. The rest of the equivalences are proved similarly:

$$\Delta\mathrm{Q}[\![\bot \overset{p}{\leftharpoondown\!\!\rightharpoonup} \bot]\!] = p\mathbf{0} + (1-p)\mathbf{0} = \mathbf{0} = \Delta\mathrm{Q}[\![\bot]\!].$$

■

Remark 5. The very last equivalence in Fig. 13 was incorrectly formulated (though never published) prior to this paper. Thanks to the formalisation developed in [13], that mistake was corrected, and all equivalences have been given a sound footing. □

5.3. Distributivity

In this section, we consider the distributivity results between the ΔQSD operators. Recall that out of the four $\mathcal{P}$ operators, three are commutative (i.e., $\bullet\!\!\rightarrow\!\!\bullet$, $\|^{\forall}$, and $\|^{\exists}$) and one is not (i.e., $\leftharpoondown\!\!\rightharpoonup$). Hence, it is only possible for right- and left-distributivity to differ when $\leftharpoondown\!\!\rightharpoonup$ is the outermost operator. That gives rise to $2 \times \binom{3}{1} + \binom{3}{1}\binom{3}{1} = 15$ possible ways for distributing $\mathcal{P}$

operators over each other. Theorem 6 establishes 3 of those 15. In our earlier work [17, §7.1], we show how the routine technique for examining the equivalence of expressions (i.e., equating the $\Delta\mathbb{Q}[\![.]\!]$ of the two sides) is not that helpful for the study of the remaining 12 distributivity results. That leads to two further developments in our earlier work [17, §7.2 and §8], which disprove the generality of 4 and 8 distributivity results using counterexamples and properisation (Theorem 8), respectively.

We use the following syntactic convention: when, in an equivalence, two $\leftrightharpoons$s are used without probability values, once on either side of the equivalence, we will assume that those probabilities are the same. For example, in the theorem below, there exists a probability p such that $o_1 \bullet\!\!\rightarrow\!\!\bullet\ (o_2 \overset{p}{\leftrightharpoons} o_3)$ is equivalent to $(o_1 \bullet\!\!\rightarrow\!\!\bullet\ o_2) \overset{p}{\leftrightharpoons} (o_1 \bullet\!\!\rightarrow\!\!\bullet\ o_3)$, but we omit the letter p from the notation.

Theorem 6. *Let* $o_1, o_2, o_3 \in \mathbb{O}$ *and* $\diamond \in \{\bullet\!\!\rightarrow\!\!\bullet, \|^\forall, \|^\exists\}$. *Then,*

- $\odot\!\odot$ *time* $\vDash o_1 \diamond (o_2 \leftrightharpoons o_3) = (o_1 \diamond o_2) \leftrightharpoons (o_1 \diamond o_3)$, *and*
- $\odot\!\odot$ *time* $\vDash (o_1 \leftrightharpoons o_2) \diamond o_3 = (o_1 \diamond o_3) \leftrightharpoons (o_2 \diamond o_3)$.

5.4. Compositional Extraction of Failure

We start the application of our properisation technique by obtaining some useful results. Theorem 7 paves the way for the applications of the above technique. They instruct one on how to accumulate failure at the rightmost corner when the operator between two pairs of parentheses is $\bullet\!\!\rightarrow\!\!\bullet$, $\leftrightharpoons$, and $\|^\forall$, respectively. Unfortunately, for $\|^\exists$ one cannot retain a simple $\|^\exists$ after the accumulation and the formula gets doubly involved: see Remark 6.

Theorem 7. *For every* $o_1, o_2, o_3 \in \mathbb{O}$,

$$(o_1 \overset{p_1}{\leftrightharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet\ (o_2 \overset{p_2}{\leftrightharpoons} \bot) = (o_1 \bullet\!\!\rightarrow\!\!\bullet\ o_2) \overset{[p_1 p_2]}{\leftrightharpoons} \bot$$
$$(o_1 \overset{p_1}{\leftrightharpoons} \bot) \overset{p}{\leftrightharpoons} (o_2 \overset{p_2}{\leftrightharpoons} \bot) = (o_1 \overset{q}{\leftrightharpoons} o_2) \overset{r}{\leftrightharpoons} \bot, \text{ where}$$
$$q = \frac{pp_1}{p_2 - pp_2 + pp_1} \text{ and } r = p_2 - pp_2 + pp_1$$
$$(o_1 \overset{p_1}{\leftrightharpoons} \bot)\|^\forall (o_2 \overset{p_2}{\leftrightharpoons} \bot) = (o_1 \|^\forall o_2) \overset{[p_1 p_2]}{\leftrightharpoons} \bot$$
$$(o_1 \overset{p_1}{\leftrightharpoons} \bot)\|^\exists (o_2 \overset{p_2}{\leftrightharpoons} \bot) = \left((o_1 \|^\exists o_2) \overset{p}{\leftrightharpoons} (o_1 \overset{q}{\leftrightharpoons} o_2)\right) \overset{r}{\leftrightharpoons} \bot, \text{ where}$$
$$p = \frac{p_1 p_2}{1-(1-p_2)(1-p_1)},\ q = \frac{p_1(1-p_2)}{p_1+p_2-2p_1p_2}, \text{ and } r = 1-(1-p_2)(1-p_1).$$

Proof. We only prove the first equivalence here. The proof is similar for the others.

By Theorems 5 and 6,

$$(o_1 \overset{p_1}{\leftrightharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet (o_2 \overset{p_2}{\leftrightharpoons} \bot) = ((o_1 \overset{p_1}{\leftrightharpoons} \bot) \bullet\!\!\rightarrow\!\!\bullet o_2) \overset{p_2}{\leftrightharpoons} \bot = ((o_1 \bullet\!\!\rightarrow\!\!\bullet o_2) \overset{p_1}{\leftrightharpoons} \bot) \overset{p_2}{\leftrightharpoons} \bot = (o_1 \bullet\!\!\rightarrow\!\!\bullet o_2)^{[p_1 p_2]} \overset{}{\leftrightharpoons} \bot.$$

■

Remark 6. Interestingly enough, there is no p such that the following holds in its full generality:

$$(o_1 \overset{p_1}{\leftrightharpoons} \bot) \|^{\exists} (o_2 \overset{p_2}{\leftrightharpoons} \bot) \overset{?}{=} (o_1 \|^{\exists} o_2) \overset{p}{\leftrightharpoons} \bot.$$

Suppose there were such a p. After some calculations one finds that equating the $\Delta\mathrm{Q}[\![.]\!]$ of the two sides implies $p = p_1 = p_2 = 1$ or $p = p_1 = p_2 = 0$. When $(o_1 \overset{p_1}{\leftrightharpoons} \bot) \|^{\exists} (o_2 \overset{p_2}{\leftrightharpoons} \bot)$ is $o_1 \|^{\exists} o_2$, in which o_1 and o_2 are being properised, that is either when $o_1 = o_2 = \top$ or $o_1 = o_2 = \bot$. □

6. Properisation: Making Failure Algebraically Accessible

Recall that the intangible mass of an outcome is a matter of semantics. In particular, it is the basic assignment (Definition 2) that determines the intangible mass of an individual outcome. Properisation is a technique that gives a syntactic representation to intangibility (as a semantic property) and then employs Theorem 7 repeatedly until the failure probability of a given complex outcome expression hits the surface. As explained before, that gives rise to a technique for swiftly calculating an outcome expression's failure probability.

Theorem 8 authorises levering properisation for a different purpose. As demonstrated in our earlier work [17, §8], properisation can also be utilised for disproving equivalences, in particular those related to distributivity of the ΔQSD operators ($\mathcal{P}$). But, before delving into the technical details, §6.1 explains why one should essentially care about properisation. §6.2 provides the technicality required for Theorem 8.

6.1. Motivation

The equivalences of Fig. 7 can be used to compute the overall failure probability of an outcome expression by transforming it so that failure ($\bot$) appears in a probabilistic choice at the top level. The examples of §3 use this technique to compute failure probabilities. But this only works if the failures are syntactically present in the outcome expression. If a primitive

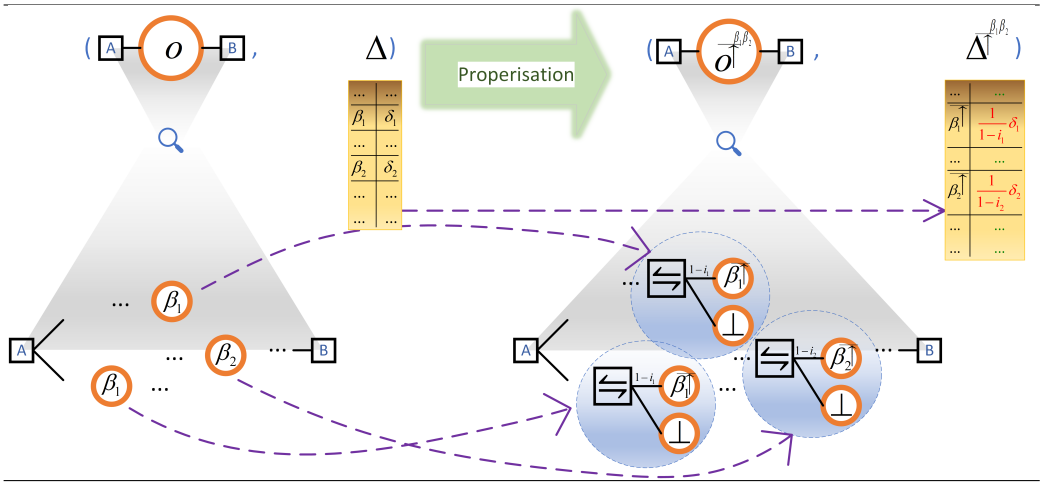


Figure 14: Properisation of β_1 and β_2 in o According to Δ

outcome is itself improper then we cannot use it. Properisation is designed to overcome this limitation. If we have an outcome, such as *main*, that is improper (nonzero failure probability) in a given basic assignment, then we rewrite it as $main\overset{\frown}{\uparrow} \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} \bot$, where "$main\overset{\frown}{\uparrow}$" now refers to *main* with a proper distribution (no failure) whilst the failure probability of $main\overset{\frown}{\uparrow} \underset{\rightharpoonup}{\overset{p}{\leftharpoondown}} \bot$ altogether is still p. The algebraic equivalences of Fig. 7 can now be used. Fig. 14 gives the big picture for a large outcome expression o. The improper outcomes β_1 and β_2, as assigned by Δ, are transformed into $\beta_1\overset{\frown}{\uparrow} \leftrightharpoons \bot$ and $\beta_2\overset{\frown}{\uparrow} \leftrightharpoons \bot$. This gives the transformed expression $o\overset{\frown}{\uparrow}^{\beta_1\beta_2}$ where the $\beta\overset{\frown}{\uparrow}$'s now refer to proper outcomes. We can use Fig. 7 on this expression to "bubble up" the failures $\bot$ to the top level.

6.2. Formulation

This section sets the stage using Theorem 8 for a technique that we call *properisation* and use for disproving potential equivalences (in their full generality).

Properisation is based on the following important observation: if two outcomes do not fail similarly, they are not equivalent. Properisation is an algebraic technique for swiftly extracting the failure behaviour of outcomes via rewriting but without assessing the rest of their timeliness behaviour. Once the failure parts of the timeliness behaviours are at hand for the two

sides, one can check whether they are equal, and if they are not, deduce that the outcomes in question are therefore unequal.

Our intuition for the choice of name "properisation" for this technique follows: recall that ΔQs are CDFs (or PDFs) of **im**proper random variables. Properisation is a technique based on making the ΔQ of an outcome o proper (by scaling it) and restoring its amount of improperness – i.e., o's intangible mass, denoted by $\Im(\Delta Q(o))$ – as a probabilistic choice (of the right weights) between o and $\bot$. That is also the intention behind the symbol we use for properisation: "$\overline{\uparrow}$." As one can see in Fig. 2a, the CDF of an improper random variable needs not to make it to the "ceiling" (i.e., 1). The symbol "$\overline{\uparrow}$" that we use is intended to resemble the act of 'sticking the CDF to the ceiling' (represented by the horizontal bar at the top of "$\overline{\uparrow}$")!

Now, the formal definitions of properisation.

Definition 4. For an $\iota \in \mathbb{I}$ such that $\Im(\iota) = i \neq 1$, write $\iota' := \iota\overline{\uparrow}$ when $dom(\iota) = dom(\iota')$ and $\iota'(x) = \frac{1}{1-i}\iota(x)$ for every $x \in dom(\iota)$. Call ι' the properisation of ι.

More on the above definition in Equation (48).

Proposition 1. $\Im(\iota\overline{\uparrow}) = 0$, *for all* $\iota \in \mathbb{I}$.

Intuitively, for IRVs, ".$\overline{\uparrow}$" produces a scaled random variable with no intangible mass.

Definition 5. Fix two basic assignments Δ, Δ' and a base variable β such that $\Delta(\beta) = \iota$. Write $\Delta' = \Delta\overline{\uparrow}^{\beta}$ when

$$\Delta'(\beta') = \Delta(\beta') \quad \text{for } \beta' \neq \beta \qquad\qquad \Delta'(\beta') = \iota\overline{\uparrow} \quad \text{otherwise.}$$

We say $\Delta\overline{\uparrow}^{\beta}$ is the result of *properisation* of β in Δ.

Intuitively, $\Delta\overline{\uparrow}^{\beta}$ produces a new basic assignment that is the same as Δ everywhere except β, where the assigned IRV is properised.

Notation 1. Write $o[o'/\beta]$ for the familiar λ-Calculus notation for substitutions: o in which every instance of β is replaced by o'.

Definition 6. Fix a basic assignment Δ and a base variable β such that $\Delta(\beta) = \iota$ where $\Im(\iota) = i$. Write $(o, \Delta)\overline{\uparrow}^{\beta} := (o', \Delta')$ when

- $o' := o[(\beta\overset{1-i}{\rightleftharpoons}\bot)/\beta]$, and
- $\Delta' := \Delta\overset{\beta}{\uparrow}$.

We say that o' is the result *properisation* of β in o according to Δ.

As a shorthand, we write $(o,\Delta)\overset{\beta_1,\beta_2}{\uparrow}$ for $\left((o,\Delta)\overset{\beta_1}{\uparrow}\right)\overset{\beta_2}{\uparrow}$ and $\Delta\overset{\beta_1,\beta_2}{\uparrow}$ for $\left(\Delta\overset{\beta_1}{\uparrow}\right)\overset{\beta_2}{\uparrow}$.

As one can see from Definition 6, the act of properisation of a base variable β in an outcome o is according to a given basic assignment Δ. That is, the move from the right-hand-side of $(o,\Delta)\overset{\beta}{\uparrow} = (o',\Delta')$ to its left-hand-side is performed by taking two steps in unison:

1. scaling according to the intangible mass of $\Delta(\beta)$ so that β is no longer improper in the resulting new basic assignment Δ'; and,
2. replacing every occurrence of β in the outcome o with the probabilistic choice that is weighted according to the intangible mass of $\Delta(\beta)$, resulting in the new outcome o'.

The idea is that the intangible mass that Δ' takes away o' returns, leaving timeliness intact. Lemma 4 utilises that idea.

Lemma 4. *Suppose that* $(o,\Delta)\overset{\beta_1,\beta_2,\ldots,\beta_n}{\uparrow} = (o',\Delta')$ *for some* $\beta_1,\beta_2,\ldots,\beta_n \in \overline{\mathbb{B}}$, $o,o' \in \mathbb{O}$ *and basic assignments* Δ *and* Δ'. *Then,* $\Delta\, Q[\![o]\!]_\Delta = \Delta\, Q[\![o']\!]_{\Delta'}$.

Theorem 8 utilises Lemma 4 for examining equivalence of pairs of outcome expressions with no properisation relationship.

Theorem 8. *Suppose* Δ *and* Δ' *are two basic assignments. Suppose also that* $o_1,o_1',o_2,o_2' \in \mathbb{O}$ *such that* $(o_1',\Delta') = (o_1,\Delta)\overset{\beta_1,\beta_2,\ldots,\beta_n}{\uparrow}$ *and* $(o_2',\Delta') = (o_2,\Delta)\overset{\beta_1,\beta_2,\ldots,\beta_n}{\uparrow}$, *for some* $\beta_1,\beta_2,\ldots,\beta_n \in \overline{\mathbb{B}}$. *Then,* $\Delta\, Q[\![o_1]\!]_\Delta = \Delta\, Q[\![o_2]\!]_\Delta$ *iff* $\Delta\, Q[\![o_1']\!]_{\Delta'} = \Delta\, Q[\![o_2']\!]_{\Delta'}$.

Even though properisation has long been used as a technique for swiftly calculating the failure rates, Theorem 8 also gives rise to a technique for disproving equivalences. That is well-detailed in our earlier work [17, §8]. Here we only outline that properisation disproving technique:

Suppose two outcome expressions o and o' the equivalence of which is to be studied. One begins by studying the equivalence of $o\overset{\beta_1,\ldots,\beta_n}{\uparrow}$ and $o'\overset{\beta_1,\ldots,\beta_n}{\uparrow}$ for some $\beta_1,\ldots,\beta_n \in \overline{\mathbb{B}}$. Now, suppose that – after the application of algebraic laws – one gets to rewrite $o\overset{\beta_1,\ldots,\beta_n}{\uparrow}$ to $(\ldots)\overset{p}{\rightleftharpoons}\bot$ and $o'\overset{\beta_1,\ldots,\beta_n}{\uparrow}$ to $(\ldots)\overset{p'}{\rightleftharpoons}\bot$. One concludes that $o \neq o'$ if one can show that $p \neq p'$.

7. Failure and Proper Probability Distributions

We have discussed that the quality attenuation ΔQ models both delay and failure of an outcome. Specifically, failure of an outcome is identified with the failure of the outcome to occur within a finite amount of time, i.e. with an infinite delay. When gauging the feasibility of a system design, one may want to do a back-of-the-envelope computation that computes only the *failure probability*, and forego a detailed analysis of the delay times; in other words, one may want to do a failure analysis.

The algebra of outcome expressions is well-suited for computing failure probabilities in a way that combines naturally with the full quality attenuation. Specifically, the failure probability is a *morphism* from the algebra of improper probability distributions to a single numeric value. The property of being a morphism simplifies the computation significantly: it implies that the failure probability of an outcome expression can be computed by combining single numerical values, and skipping over the intermediate computation of the full quality attenuation. We will present an example shortly.

In addition, the failure probability also sheds light on the full quality attenuation. One may decompose a quality attenuation into a pair of a proper probability distribution (some finite delay) and a probability of failure (infinite delay). But, it turns out that the components of this pair are not independent. Specifically, the operation $\|^{\exists}$ combines them in a nontrivial way. By computing with expressions in the combined algebra of improper probability distributions, this dependence is automatically taken into account. The decomposition of an expression into a proper probability distribution and a failure part is what we refer to as *properisation*, introduced in §6.

In §7.1, we will define the algebra of failure probabilities $\mathbb{F}$, the failure probability morphism $F : \mathbb{I} \to \mathbb{F}$, and present a computation for the cache example from §3. In §7.2, we will describe the decomposition of improper probability distributions into a proper probability distribution and a probability of failure. In §7.3, we will conceive the decomposition as an isomorphism of algebras $\mathbb{I} \cong \mathbb{P} \rtimes \mathbb{F}$, where $\mathbb{P}$ is the algebra of proper probability distributions. Here, the construction $\rtimes$ denotes pairing, but also takes into account that the operation $\|^{\exists}$ mixes the components of the pair.

7.1. Failure Probability Morphism

In order to introduce the failure probability morphism, let us revisit the cache example from the introduction. Specifically, we consider the outcome

expression (Eq. (1))

$$c\text{-}hit \overset{95\%}{\leftrightharpoons} (c\text{-}miss \bullet\!\!\rightarrow\!\!\bullet\ main). \tag{25}$$

We had assumed that the memory read *main* can fail with the very small chance of 10^{-16}. Let us denote this fact by the notation $\tilde{F}(main) = 10^{-16}$, where the function $\tilde{F} : \mathbb{O} \to \mathbb{F}$ maps an outcome expression to its probability of failure. Here, $\mathbb{F}$ denotes the algebra of failure probabilities: each element $p \in \mathbb{F}$ is a numerical probability, $p \in [0, 1]$, but we use the emboldened notation $\mathbb{F}$ to highlight that these probabilities can be combined with the following operations:

Definition 7 (Algebra of Failure Probabilities)**.** The algebra $\mathbb{F}$ of failure probabilities consists of the carrier set $[0, 1]$ and the operations $\{\bullet\!\!\rightarrow\!\!\bullet, \leftrightharpoons, \|^{\exists}, \|^{\forall}\} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ defined as follows:

$$p_1 \bullet\!\!\rightarrow\!\!\bullet\ p_2 \ := \ p_1 + p_2 - p_1 p_2, \tag{26}$$

$$p_1 \overset{p}{\leftrightharpoons} p_2 \ := \ pp_1 + (1-p)p_2, \tag{27}$$

$$p_1 \ \|^{\forall}\ p_2 \ := \ p_1 + p_2 - p_1 p_2, \tag{28}$$

$$p_1 \|^{\exists}\ p_2 \ := \ p_1 p_2. \tag{29}$$

Now, a *morphism* between algebras denotes a map that is compatible with the operations of the algebra; in our case, a morphism between algebras $\mathbb{A}, \mathbb{B}$ is a map $f : \mathbb{A} \to \mathbb{B}$ such that $f(x \diamond y) = f(x) \diamond f(y)$ where $\diamond$ is a placeholder for any of the operations $\bullet\!\!\rightarrow\!\!\bullet, \leftrightharpoons, \|^{\forall}$, or $\|^{\exists}$. In other words, it makes no difference whether we first combine elements from the algebra $\mathbb{A}$ and then map them using f, or whether we first map them to the algebra $\mathbb{B}$ and then combine them.

We claim that the map $\tilde{F}$ from outcome expressions to failure probabilities introduced above is a morphism. Assuming this for a moment, we can compute the failure probability by exchanging morphism and operations

repeatedly:

$$
\begin{aligned}
&\tilde{F}\left(\textit{c-hit} \overset{95\%}{\rightleftharpoons} (\textit{c-miss} \bullet\!\!\rightarrow\!\!\bullet\ \textit{main})\right) \\
&= \tilde{F}\left(\textit{c-hit}\right) \overset{95\%}{\rightleftharpoons} \tilde{F}\left(\textit{c-miss} \bullet\!\!\rightarrow\!\!\bullet\ \textit{main}\right) && (30)\\
&= 0 \overset{95\%}{\rightleftharpoons} \tilde{F}\left(\textit{c-miss} \bullet\!\!\rightarrow\!\!\bullet\ \textit{main}\right) && (31)\\
&= 5\% \cdot \tilde{F}\left(\textit{c-miss} \bullet\!\!\rightarrow\!\!\bullet\ \textit{main}\right) && (32)\\
&= 5\% \cdot \left(\tilde{F}\left(\textit{c-miss}\right) \bullet\!\!\rightarrow\!\!\bullet\ \tilde{F}\left(\textit{main}\right)\right) && (33)\\
&= 5\% \cdot \left(0 \bullet\!\!\rightarrow\!\!\bullet\ \tilde{F}\left(\textit{main}\right)\right) && (34)\\
&= 5\% \cdot \tilde{F}\left(\textit{main}\right) && (35)\\
&= 5\% \cdot 10^{-16} = 5 \cdot 10^{-18}. && (36)
\end{aligned}
$$

In addition to the morphism property and the rules for combining failure probabilities, we have used the fact that the outcomes *c-miss* and *c-hit* always succeed, and the assumption $\tilde{F}(\textit{main}) = 10^{-16}$ noted above. This example highlights how the property of being a morphism simplifies computations. In addition, by focusing our attention on outcome expressions and applying simplifications only while computing quantities of interest from it, we can revisit those simplifications later in the design. For example, the above computation can be revisited at a later stage to account for an imperfect cache that does not satisfy $\tilde{F}(\textit{c-miss}) = 0$.

Motivated by the above example, we now define the mapping $\tilde{F}$ more precisely, relate it to the full quality attenuation, and prove that it is a morphism.

We have applied the mapping $\tilde{F} : \mathbb{O} \to \mathbb{F}$ directly to outcome expressions. However, from an algebraic perspective, it is more convenient to define it as a composition $\tilde{F} := F \circ \Delta Q$ of the quality attenuation mapping $\Delta Q : \mathbb{O} \to \mathbb{I}$ and a mapping $F : \mathbb{I} \to \mathbb{F}$ which maps improper probability distributions to their probability of failure:

Definition 8. We define the *failure probability* to be the mapping $F : \mathbb{I} \to \mathbb{F}$ given by

$$F(Q) := 1 - \lim_{t \to \infty} Q(t),$$

where $Q(t)$ is the cumulative distribution function of the element $Q \in \mathbb{I}$.

Lemma 5. *The failure probability $F : \mathbb{I} \to \mathbb{F}$ is a morphism of algebras.*

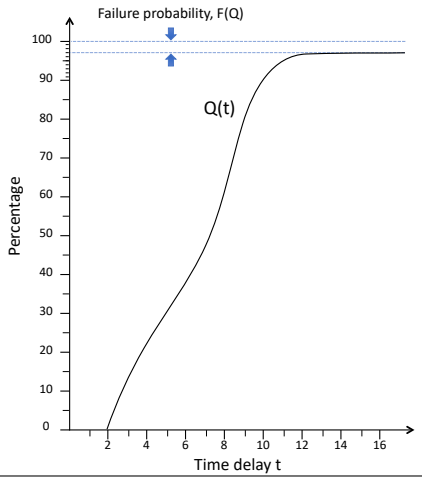


Figure 15: The failure probability $F(Q)$ is the probability that the outcome never occurs, i.e. it is the difference between the limit of the cumulative probability $Q(t)$ as $t \to \infty$ and perfect 100% probability.

Proof. For the operations $\leftrightharpoons, \|^{\forall}, \|^{\exists}$, this follows from the fact that limits commute with arithmetic operations. For example, by using the definition of $\|^{\exists}$ for improper probability distributions $Q_1, Q_2 \in \mathbb{I}$, we find

$$\begin{aligned} F(Q_1 \parallel^{\exists} Q_2) &= 1 - \lim_{t\to\infty} (Q_1 \parallel^{\exists} Q_2)(t) && (37)\\ &= 1 - \lim_{t\to\infty} [Q_1(t) + Q_2(t) - Q_1(t) \times Q_2(t)] && (38)\\ &= 1 - \lim_{t\to\infty} [1 - (1 - Q_1(t)) \times (1 - Q_2(t))] && (39)\\ &= 1 - [1 - F(Q_1) \times F(Q_2)] && (40)\\ &= F(Q_1) \parallel^{\exists} F(Q_1). && (41) \end{aligned}$$

Here, Eq. (38) uses the definition of $\|^{\exists}$, and Eq. (40), uses that $\lim_{t\to\infty}$ commutes with $+$ and $\times$.

For sequential composition, $\bullet\!\!\rightarrow\!\!\bullet$, the result makes sense intuitively: A sequence of operations succeeds if and only if both of the operations succeed, so the total success probability should be the product of the individual success probabilities. For a formal proof of this fact, we use Lemma 6 in §Appendix

A.3. We have

$$\begin{aligned} F(Q_1 \bullet\!\!\rightarrow\!\!\bullet\, Q_2) &= 1 - \lim_{t\to\infty} (Q_1 \bullet\!\!\rightarrow\!\!\bullet\, Q_2)(t) && (42)\\ &= 1 - \lim_{t\to\infty} (Q_1 * Q_2)(t) && (43)\\ &= 1 - \left(\lim_{t\to\infty} Q_1(t)\right)\left(\lim_{t\to\infty} Q_2(t)\right) && (44)\\ &= 1 - (1 - F(Q_1)) \times (1 - F(Q_2)) && (45)\\ &= F(Q_1) + F(Q_2) - F(Q_1)F(Q_2) && (46)\\ &= F(Q_1) \bullet\!\!\rightarrow\!\!\bullet\, F(Q_2). && (47) \end{aligned}$$

Here, we have used Lemma 6 in Eq. (44). ■

7.2. Decomposition into Failure and Proper Probability Distribution

We have discussed quality attenuation ΔQ in terms of improper probability distributions. An improper probability distribution $P \in \mathbb{I}$ assigns a probability $P(t)$ to the event that an outcome occurs with a delay $\leq t$; but, it may also assign a non-zero probability to the event that the outcome never occurs (within an acceptable time frame), so that the cumulative weight $\lim_{t\to\infty} P(t)$ can be strictly less than one.

However, any improper probability distribution $P \in \mathbb{I}$ can also be described by its probability of failure $F(P)$ and a *conditional probability*, namely the probability $\tilde{P}(t)$ that the outcome occurs with a delay $\leq t$ on the condition that the outcome does occur at all. This conditional probability can be obtained by scaling the cumulative distribution function $\tilde{P}(t) := P(t)/\left(\lim_{\tau\to\infty} P(\tau)\right)$, except for the corner case where the outcome has zero chance of occurring, i.e., where the denominator vanishes.

Properisation gives this decomposition a syntactic account. Specifically, in line with Definition 4, one can write every improper probability distribution $P \in \mathbb{I}$ that does not always fail as a probabilistic choice between a *proper* probability distribution $\tilde{P}$ and failure $\bot$:

$$P = \tilde{P} \overset{[1-p]}{\leftharpoondown\!\!\rightharpoonup} \bot, \qquad \text{where } p = F(P),\ \tilde{P}(t) = \frac{P(t)}{(1-p)},\ \text{provided that } p \neq 1. \tag{48}$$

For the corner case where the outcome always fails, we have to write $P = \bot$ instead.

This method of expressing a quality attenuation is helpful when one wants to compute the failure probability whilst *retaining information* about finite

delay times. Specifically, combining two expressions of the form given in Eq. (48) with one of the operations will yield another expression of the same form. This is straightforward for $\bullet\!\!\rightarrow\!\!\bullet$ and $\|^{\forall}$, but the expression is more complicated in the case of $\|^{\exists}$. The formulas are presented in Theorem 7.

7.3. Decomposition of Algebras

It is instructive to recast the above observation that every improper probability distribution can be decomposed into a probability of failure and a proper probability distribution in terms of algebras and their morphisms, i.e., in terms of commutative diagrams, which are a common tool in abstract algebra and category theory.

First, we define the subset of proper probability distributions by

$$\mathbb{P} := \{Q \in \mathbb{I} : F(Q) = 0\} \subset \mathbb{I}. \tag{49}$$

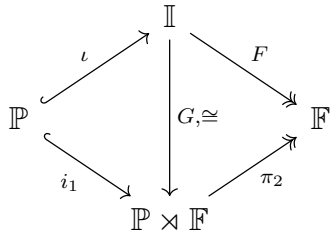
We can check that this set is, in fact, closed under the algebra operations, and, hence, a subalgebra of $\mathbb{I}$.

The relation between proper probability distributions $\mathbb{P}$, the improper probability distributions $\mathbb{I}$, and the algebra of failure probabilities $\mathbb{F}$ can be summarised in the diagram

$$\mathbb{P} \xhookrightarrow{\iota} \mathbb{I} \xrightarrow{F} \!\!\!\!\rightarrow\; \mathbb{F} \ . \tag{50}$$

Here, ι is the identity embedding, which maps any cumulative distribution function $P \in \mathbb{P}$ to the function $\iota(P)(t) := P(t)$ representing a (potentially) improper probability distribution. The hook on the arrow indicates that this morphism is injective. The morphism F is the failure probability defined as before; the double arrowhead indicates that this morphism is surjective. In addition, the maps satisfy $F \circ \iota = 0$.

The observation that any improper probability distribution can be written as pair of a proper probability distribution and a failure probability can be expressed in a commutative diagram



(51)

which we now explain. The algebra $\mathbb{P} \rtimes \mathbb{F}$ will be defined properly below, but the intuition is that its carrier set is roughly the set of pairs $\sim \mathbb{P} \times \mathbb{F}$ composed of a proper probability distribution and a failure probability. The morphism G in the middle maps from the algebra $\mathbb{I}$ to this algebra, and the symbol $\cong$ indicates that this morphism is, in fact, an isomorphism, i.e. that it is a bijection between the elements of both algebras that preserves the operations. The intuition is that the isomorphism G lifts Eq. (48) from a convenient rule for computation to a bona-fide map between algebras. This diagram is *commutative*, which means that the composition of arrows along any sequence with the same starting and ending point gives the same map. For example, commutativity of the left triangle amounts to the claim that $G \circ \iota = i_1$, and the right triangle amounts to $F = \pi_2 \circ G$. In addition to the fact that the algebras in the middle are isomorphic, these identities tell us that the isomorphism relates the embedding ι to the embedding i_1, and the surjection F to the surjection π_2.

We now define the carrier set of the object $\mathbb{P} \rtimes \mathbb{F}$, as well as the maps i_1 and π_2. The carrier set is given by the set $(\mathbb{P} \times [0, 1)) \cup \{\bot\}$, i.e. a set of pairs adjoined with a formal failure element $\bot$. The map i_1 is defined as $i_1(P) := (P, 0)$. The map π_2 is defined as $\pi_2(P, p) = p$ on a pair, and $\pi_2(\bot) = 1$ for failure. It is straightforward to see that $\pi_2 \circ i_1 = 0$, which implies that the outermost paths of the diagram commute.

Next, we define the map $G : \mathbb{I} \to \mathbb{P} \rtimes \mathbb{F}$ on the carrier sets:

$$G(P) := \begin{cases} (P(t)/(1 - F(P)), F(P)) & \text{if } F(P) \neq 1 \\ \bot & \text{otherwise.} \end{cases} \tag{52}$$

Here, $P(t)$ is the cumulative distribution function of the improper probability distribution $P \in \mathbb{I}$. It is straightforward to check that the map $\tilde{G} : \mathbb{P} \rtimes \mathbb{F} \to \mathbb{I}$ defined by

$$\tilde{G}(\tilde{P}, p) := \tilde{P} \overset{[1-p]}{\rightharpoonup} \bot, \quad \tilde{G}(\bot) := \bot \tag{53}$$

is the inverse of the map G, i.e. that both $G \circ \tilde{G}$ and $\tilde{G} \circ G$ are the identity maps on the respective carrier sets. This shows that G is a bijection. With these definitions, it is also straightforward to check that the left and right triangles of the diagram are commutative, $G \circ \iota = i_1$, and $F = \pi_2 \circ G$.

We still need to give $\mathbb{P} \rtimes \mathbb{F}$ algebraic structure, and show that G, i_1, π_2 are not just maps between sets, but morphisms between algebras.

Definition 9. (Algebra $\mathbb{P} \rtimes \mathbb{F}$) On the object $\mathbb{P} \rtimes \mathbb{F}$ with the carrier set $(\mathbb{P} \times [0,1)) \cup \{\bot\}$, we define the algebra operations as follows: When both arguments of the operation are pairs, we define

$$
\begin{aligned}
(\tilde{P}_1, p_1) \bullet\!\!\rightarrow\!\!\bullet\, (\tilde{P}_2, p_2) &:= (\tilde{P}_1 \bullet\!\!\rightarrow\!\!\bullet\, \tilde{P}_2,\ p_1 \bullet\!\!\rightarrow\!\!\bullet\, p_2), && (54)\\
(\tilde{P}_1, p_1) \overset{\alpha}{\leftrightharpoons} (\tilde{P}_2, p_2) &:= (\tilde{P}_1 \overset{r}{\leftrightharpoons} \tilde{P}_2,\ p_1 \overset{\alpha}{\leftrightharpoons} p_2) && (55)\\
&\quad \text{where } r = \frac{\alpha p_1}{(1-\alpha)p_1 + \alpha p_2}, && (56)\\
(\tilde{P}_1, p_1) \parallel^{\forall} (\tilde{P}_2, p_2) &:= (\tilde{P}_1 \parallel^{\forall} \tilde{P}_2,\ p_1 \parallel^{\forall} p_2), && (57)\\
(\tilde{P}_1, p_1) \parallel^{\exists} (\tilde{P}_2, p_2) &:= ((\tilde{P}_1 \parallel^{\exists} \tilde{P}_2) \overset{[p]}{\leftrightharpoons} (\tilde{P}_1 \overset{[q]}{\leftrightharpoons} \tilde{P}_2),\ p_1 \parallel^{\exists} p_2) && (58)\\
&\quad \text{where } p = \frac{p_1 p_2}{1-(1-p_2)(1-p_1)}, && (59)\\
&\quad \text{and } q = \frac{p_1(1-p_2)}{p_1 + p_2 - 2p_1 p_2}. && (60)
\end{aligned}
$$

For the case where the first argument is $\bot$, we define

$$
\begin{aligned}
\bot \bullet\!\!\rightarrow\!\!\bullet\, (\tilde{P}_2, p_2) &:= \bot, && (61)\\
\bot \overset{\alpha}{\leftrightharpoons} (\tilde{P}_2, p_2) &:= (\tilde{P}_2,\ 1 \overset{\alpha}{\leftrightharpoons} p_2), && (62)\\
\bot \parallel^{\forall} (\tilde{P}_2, p_2) &:= \bot, && (63)\\
\bot \parallel^{\exists} (\tilde{P}_2, p_2) &:= (\tilde{P}_2, p_2), && (64)
\end{aligned}
$$

For the case where the second argument is $\bot$, we define

$$
\begin{aligned}
(\tilde{P}_1, p_1) \bullet\!\!\rightarrow\!\!\bullet\, \bot &:= \bot, && (65)\\
(\tilde{P}_1, p_1) \overset{\alpha}{\leftrightharpoons} \bot &:= (\tilde{P}_1,\ p_1 \overset{\alpha}{\leftrightharpoons} 1), && (66)\\
(\tilde{P}_1, p_1) \parallel^{\forall} \bot &:= \bot, && (67)\\
(\tilde{P}_1, p_1) \parallel^{\exists} \bot &:= (\tilde{P}_1, p_1). && (68)
\end{aligned}
$$

Finally, we define the result to $\bot$ whenever both arguments are equal to $\bot$.

The definitions of the operations of the algebra $\mathbb{P} \rtimes \mathbb{F}$ were chosen in a way such that the map $\tilde{G}$ becomes a morphism of algebras. This fact is the most important claim of the commutative diagram above, but it is merely another way of presenting Theorem 7.

We note that some operations, such as $\bullet\!\!\rightarrow\!\!\bullet$ and $\parallel^{\forall}$, are performed by applying them to each component of the pair separately. However, for the operations $\leftrightharpoons$ and $\parallel^{\exists}$, the first component is more complicated, and contains

information from the second component. In other words, when computing the probability distribution of finite delay times, the probability of failure has to be taken into account in the computation, it would not be correct to treat them in isolation! The asymmetric product symbol $\rtimes$ was chosen to highlight this fact.

The other maps are morphisms as well. To see this, note that G is the inverse of the morphism $\tilde{G}$, hence it is also a morphism. The maps i_1 and π_2 are morphisms, because we have already shown that the diagram commutes as a diagram of maps between sets, and these maps can be written as compositions of morphisms: $i_1 = G \circ \iota$ and $\pi_2 = F \circ \tilde{G}$.

Even though the isomorphism $\mathbb{I} \cong \mathbb{P} \rtimes \mathbb{F}$ clarifies the algebraic relations between failure and quality attenuation, for practical computations, we do not advise to use the pair notation used to define $\mathbb{P} \rtimes \mathbb{F}$. Instead, we recommend to work in the algebra $\mathbb{I}$ and use the expression presented in Eq. (48). We note that this expression makes use of the fact that any proper probability distribution $\tilde{P}$ is also a valid improper probability distribution, i.e. that the embedding ι is transparent as far as notation is concerned.

8. Related Work

ΔQSD has been used in practice by a small group of practitioners for a couple of decades now [8, 9, 10, 11, 12]. The first formalisation of ΔQSD was, however, done quite recently by Haeri et al. [13]. We use that formalisation as a foundation.

Teigen et al [11] use ΔQ to develop a novel model of WiFi performance that produces complete latency distributions. The model is validated by comparison with previous modeling work and real-world measurements. It would be very interesting to apply ΔQSD to an outcome description of the protocol to see if this can replicate the same results.

Elsewhere, Gajda [18] attempts to model latency distributions but allows operations that do not preserve total probability, hence, leading to incorrect conclusions about failure probabilities.

Business Process Modelling and Notation (BPMN) [19] is a diagram scheme which is closely related to Outcome Diagrams (although with some details that are not considered relevant to ΔQSD). BPMN supports all ΔQSD operators except probabilistic choice. The closest operator is their "xor" gateway, which is essentially $\overset{[0.5]}{\rightleftharpoons}$. It is less expressive to the extent that it makes

it impossible to consider systems such as the example in §3. Of the attempts for formalising BPMN, those of Wong and Gibbons [20, 21] are the most related to our work. Wong and Gibbons use the CSP process algebra for that purpose and further develop it to enable the specification of timing constraints on concurrent systems. Their developments allow mechanical verification of behavioural properties of BPMN diagrams using the FDR2 [22] refinement checker. Whilst Wong and Gibbons prove many interesting properties of their BPMN instances, they do not consider algebraic equivalences or algebraic structures for BPMN as we do in this work for ΔQSD. A less related BPMN formalisation work is that of El Hichami et al. [23], which provides a denotational semantics based on the Max+ algebra as an execution model for BPMN. They list a handful of algebraic equivalences in Max+ only axiomatically, and make no attempt to study the equivalence of BPMN diagrams based on their Max+ semantics.

When it comes to timeliness analysis, an important advantage of outcome diagrams over BPMNs is Definition 3, which formally defines the timeliness analysis of outcome diagrams. Definition 3 is fundamental to the applicability of the model theory we employ in this paper (§4.3). We are not aware of any formally defined recipe for timeliness analysis of BPMNs. The closest attempts that we could find are the following two: Friedenstab et al. [24] borrow constructs from Business Activity Monitoring [25] to augment BPMN with a graphical notation for describing certain timeliness matters; and Morales [26] informally describes how to transform BPMN diagrams to timed automata networks, suggesting qualitative analysis of timeliness.

Performance Evaluation Process Algebra (PEPA) [27] is an algebraic language for performance modelling of systems. PEPA is successful and well-published with a rich family of formalisations with various interesting theoretical properties. However, PEPA suffers from several shortcomings that make it difficult to apply to real-world software systems. For example, PEPA does not model open or partially-specified systems; every detail of the system needs to be determined in advance. Since PEPA does not allow goals and objectives to be specified, it offers no assistance when comparing the predicted performance with the requirements. PEPA also suffers from state explosion, rapidly making it impractical, although more recent PEPA technology employs continuous approximations of the states to contain some of the state explosion. This is similar to the use of IRVs in ΔQSD but rather *ad hoc* in comparison. Less conservative alternatives to PEPA like SCEL [28] allow open systems but suffer from even more state explosion. CARMA [29]

addresses a lot of the problems with PEPA, using a fluid approximation to manage the state explosion.

PerformERL [30] is an Erlang toolset, which focuses on monitoring the relationship between load repeatability and internal resource allocation. The authors advertise their toolset as an assistant for making early stage performance decisions, but it is unclear how it does this. Unlike ΔQSD, monitoring (like testing) requires implementation of the system specification up to a certain level. The closer the implementation is to the full specification, the more reliable the monitoring will become, but the analysis is then no longer early-stage. Less accurate monitoring, on the other hand, is not reliable for decision making. The closest PerformERL gets to this paper is its lightweight theoretical work out of the monitoring overhead it imposes to the system under development.

Finally, Failure Modes Effects Analysis [31] (FMEA) considers how failures propagate through a system but, unlike ΔQSD, does not model delays. We are not aware of any formalisation of FMEA that can serve algebraic developments like those on failure in this paper.

9. Conclusion and Future Work

The ΔQSD paradigm defines systems as directed graphs, called outcome expressions, that give the causal connections between all of the system's behaviours. Outcome expressions consist of primitive outcomes and four compositional operators, namely sequential composition, probabilistic choice, first-to-finish synchronisation, and last-to-finish synchronisation. The main theoretical contribution of this paper is to define and prove algebraic equivalence relations between outcome expressions (see §4 and §5). We define and prove correctness of an extensive set of equivalences with respect to timeliness, where two outcome expressions are equivalent when they have identical timeliness (see Fig. 7). To show how this works in practice, we give a case study of a memory system consisting of a local cache, networked memory read with timeout, and the ability to retry (see §2 and §3). The design was successively refined, and the compositional properties of the algebra used to rapidly extract key performance features. We are developing software tool support for ΔQSD and the equivalences will allow considerable simplification of the computations required to compute timeliness. The second theoretical contribution of the paper is an algebraic study of failure. We show how to

make failure visible in outcome expressions and we give a general algebraic model that combines failure and delay in a single model (see §6 and §7).

The ΔQSD paradigm has been developed and used to resolve issues in production systems for more than two decades. As a diagnostic tool, it provides reasoned expectations of operational behaviour based on observable events. When observations conflict with these expectations, ΔQSD can be used to isolate the cause, which might be a design flaw, runtime violation of invariants or incorrect observations. As a design tool, it enables issues of resource consumption, in particular timeliness, to be considered from the very beginning of the design process, and engineering trade-offs to be explored early.

More generally, ΔQSD provides a language for expressing issues of uncertainty and risk for a range of stakeholders; having this language formalised and supported by software tools removes subjectivity and increases confidence in the conclusions. Uncertainty in the design phase will relate to unknowns of component performance and design details, whereas in an operational system, it will be due to variations in performance and loading. Having a single formalism that can express all of this using probability distributions increases the value of the initial design capture. Experience suggests that simply asking the questions required to define timeliness and construct an outcome diagram will expose many misconceptions and potential design flaws. One can only speculate how different the deployment of the Horizon accounting system in the British Post Office [32] might have been if such a methodology had been employed.

Future work will extend the formalism to other types of resources, of various kinds: ephemeral resources such as CPU cycles, power and interface capacity; static resources such as memory; and 'level' resources such as energy. These factor directly into system costs and performance, and the trade-offs between these. Having a robust formalism for exploring such trade-offs throughout the system life-cycle will enable system designers (and their management) to have more confidence in the design process and help many blind alleys to be avoided.

The formalism developed in this paper is immediately applicable to real engineering tasks, and we look forward to expanding its scope with the consideration of more types of resources and software tool support.

Acknowledgements

This research is funded by IOG as a part of an ongoing project for incorporating performance as a first-class factor of the software development life cycle. When the routine proof technique did not work for distributivity, Andre Knispel (of IOG) suggested that we could utilise easier properties to obtain the disproofs using contrapositive reasoning. We would like to thank him for that suggestion.

Appendix A. Improper Probability Distributions, and Measure Theory

In this appendix, we give a mathematically complete definition of the space $\mathbb{I}$ of improper probability distributions, also known as defective distributions [16]. We want to cover both discrete and continuous distributions simultaneously, for which we need measure theory as presented in [33].

Appendix A.1. Carrier Set $\mathbb{I}$

To define $\mathbb{I}$, we consider the measurable space $([0,\infty),\mathcal{B})$ of the half-line together with the σ-algebra $\mathcal{B}$ of Borel sets. Then, we define the space $\mathbb{I}$ to be the set of measures whose total weight is bounded by one:

$$\mathbb{I} := \{P : P \text{ measure on } ([0,\infty),\mathcal{B}), \quad P([0,\infty)) \leq 1\}. \tag{A.1}$$

Remember that a measure P assigns, to each Borel set A, a number $P(A)$. A measure $P \in \mathbb{I}$ is a *proper* probability distribution if $P([0,\infty)) = 1$.

For practical computations, such measures are best represented in terms of their *cumulative distribution functions* (CDFs). For every measure $P \in \mathbb{I}$, we define its cumulative distribution function

$$F_P : [0,\infty) \to [0,1], \quad F_P(t) := P([0,t]). \tag{A.2}$$

This function is increasing, $t_1 \leq t_2$ implies $F_P(t_1) \leq F_P(t_2)$, and it is right-continuous, $\lim_{t>t_0, t\to t_0} F_P(t) = F_P(t_0)$. Since $P([0,\infty)]) \leq 1$, this function is bounded by $F_P(t) \in [0,1]$.

Conversely, using Theorem 3.5 in [33], for every function $F : [0,\infty) \to [0,1]$ which is increasing and right-continuous, there exists a measure P with $P((a,b]) = F(b) - F(a)$. The condition $F(t) \leq 1$ implies $P((0,\infty)) \leq 1$. As $F(0) \geq 0$, we can set $P(\{0\}) := F(0)$; this uniquely defines the measure P. With this definition, F is also the CDF of P.

To summarize, the space $\mathbb{I}$ is in bijection with the space of functions $[0,\infty) \to [0,1]$ that are increasing and right-continuous.

Appendix A.2. Operations on $\mathbb{I}$

We now define four operations that map $\mathbb{I} \times \mathbb{I} \to \mathbb{I}$: probabilistic choice, first-to-finish, last-to-finish, and convolution.

For any two measures $P_1, P_2 \in \mathbb{I}$, we define their *probabilistic choice* as

$$\left(P_1 \stackrel{m}{\underset{m'}{+}} P_2\right)(A) := \frac{m}{m+m'} P_1(A) + \frac{m'}{m+m'} P_2(A) \tag{A.3}$$

for all Borel sets A. It is straightforward to show that this definition gives again a measure in $\mathbb{I}$.

For any two measures $P_1, P_2 \in \mathbb{I}$ with cumulative distribution functions F_1, F_2, we define the *last-to-finish* $P_1 \parallel^{\forall} P_2$ to be the measure $P_3 \in \mathbb{I}$ whose CDF F_3 is given by

$$F_3(t) := F_1(t) \cdot F_2(t), \tag{A.4}$$

for all $t \in [0, \infty)$. It is straightforward to check that F_3 is increasing, right-continuous and maps to $[0, 1]$.

Similarly, we define *first-to-finish* $P_1 \parallel^{\exists} P_2$ to be the measure $P_3 \in \mathbb{I}$ whose CDF F_3 is given by

$$F_3(t) := F_1(t) + F_2(t) - F_1(t) \times F_2(t). \tag{A.5}$$

It is straightforward to check that F_3 right-continuous. To see that it is increasing and bounded, use the identity $p + q - pq = 1 - (1 - p)(1 - q)$.

Finally, we define the *convolution* $P_1 * P_2$ of two measures $P_1, P_2 \in \mathbb{I}$ to be the measure P_3 given by

$$P_3(A) := \int_{[0,\infty)\times[0,\infty)} \chi_A(t_1 + t_2)\, d(P_1 \times P_2). \tag{A.6}$$

Here, χ_A denotes the characteristic function of the Borel set A, that is $\chi_A(t) = 1$ if $t \in A$ and $\chi_A(t) = 0$ if $t \notin A$. Using the monotone convergence theorem, it can be shown that this definition satisfies σ-additivity and hence gives a measure. We note that

$$(P_1 * P_2)([0, \infty)) = P_1([0, \infty)) \cdot P_2([0, \infty)), \tag{A.7}$$

because

$$\begin{aligned} P_3([0, \infty)) &= \int_{[0,\infty)\times[0,\infty)} \chi_{[0,\infty)}(t_1 + t_2)\, d(P_1 \times P_2) && \text{(A.8)} \\ &= \int_{[0,\infty)\times[0,\infty)} 1\, d(P_1 \times P_2) && \text{(A.9)} \\ &= P_1([0, \infty)) \cdot P_2([0, \infty)). && \text{(A.10)} \end{aligned}$$

This shows that P_3 is bounded by one, as P_1, P_2 are assumed to be bounded by one. These considerations imply that $P_3 \in \mathbb{I}$.

Appendix A.3. Properties of Convolution

In the main text, we have claimed that the failure probability is a morphism from improper probability distributions to the algebra of failure probabilities. In order to show that the failure probability commutes with sequential composition, we require the following

Lemma 6 (Success probability of the convolution)**.** *Let* $Q_1, Q_2 \in \mathbb{I}$*. Then,*

$$\lim_{t\to\infty}(Q_1 * Q_2)(t) = \left(\lim_{t\to\infty} Q_1(t)\right) \times \left(\lim_{t\to\infty} Q_2(t)\right). \tag{A.11}$$

Proof. Denote $Q_3 = Q_1 * Q_2$. The limits of the cumulative distributions functions can be expressed in terms of the measures as $\lim_{t\to\infty} Q_j(t) = Q_j([0,\infty))$ for $j = 1, 2, 3$. But then, the claim is the same as Eq. (A.7). ■

References

[1] Predictable Network Solutions Ltd (PNSol) (2022). [link].
URL http://www.pnsol.com

[2] J. T. Bradley, Towards reliable modelling with stochastic process algebras, Ph.D. thesis, University of Bristol, Department of Computer Science (October 1999).

[3] D. C. Reeve, A New Blueprint for Network QoS, Ph.D. thesis, Computing Laboratory, University of Kent, Canterbury, Kent, UK (August 2003).
URL http://www.cs.kent.ac.uk/pubs/2003/1892

[4] R. Beuran, M. Ivanovici, B. Dobinson, Network quality of service measurement system for application requirements evaluation, in: International Symposium on Performance Evaluation of Computer and Telecommunication Systems, SPECTS'03, 2003, pp. 380–387.

[5] L. Leahu, Analysis and predictive modeling of the performance of the atlas tdaq network, Ph.D. thesis, Bucharest, Tech. U. (January 2013).

[6] S. L. Gaixas, J. Perelló, D. Careglio, E. Grasa, M. Tarzan, N. Davies, P. Thompson, Assuring QoS guarantees for heterogeneous services in RINA networks with ΔQ, in: 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 584–589. doi:10.1109/CloudCom.2016.0101.

[7] P. Thompson, N. Davies, Towards a performance management architecture for large-scale distributed systems using rina, in: 2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2020, pp. 29–34.

[8] P. Thompson, R. Hernadaz, Quality attenuation measurement architecture and requirements, Tech. Rep. TR-452.1, Broadband Forum (September 2020).
URL https://www.broadband-forum.org/download/TR-452.1.pdf

[9] P. Thompson, Tr-452.2 quality attenuation measurements using active test protocols, Tech. rep., The Broadband Forum (November 2022).

[10] N. Davies, P. Thompson, G. Young, J. Newton, B. Teigen, M. Olden, Measuring network impact on application outcomes using quality attenuation, in: Measuring Network Quality for End-Users, Internet Architecture Board, 2021, pp. 43–52.
URL https://shorturl.at/bvxGT

[11] B. Ivar Teigen, N. Davies, K. Olav Ellefsen, T. Skeie, J. Torresen, Quantifying the Quality Attenuation of WiFi, in: S. Oteafy, E. Bulut, F. Tschorsch (Eds.), IEEE 47^{th} LCN, IEEE, 2022, pp. 189–197. doi:10.1109/LCN53696.2022.9843690.

[12] D. Coutts, N. Davies, M. Szamotulski, P. Thompson, Introduction to the Design of the Data Diffusion and Networking for Cardano Shelley, Tech. rep., IOHK (August 2020).
URL https://hydra.iohk.io/build/20405228/download/1/network-design.pdf

[13] S. H. Haeri, P. Thompson, N. Davies, P. Van Roy, K. Hammond, J. Chapman, Mind Your Outcomes: The ΔQSD Paradigm for Quality-Centric Systems Development and Its Application to a Blockchain Case

Study, Computers 11 (3) (2022) 45. doi:10.3390/computers11030045.
URL https://www.mdpi.com/2073-431X/11/3/45

[14] P. Van Roy, N. Davies, P. Thompson, S. H. Haeri, ΔQSD: Designing systems with predictable latency at high load, Tutorial, HiPEAC 2023 (Conf. High Perf. Emb. Arch. & Compil.) (January 2023).
URL shorturl.at/dmKSW

[15] S. H. Haeri, P. W. Thompson, P. Van Roy, M. Haveraaen, N. J. Davies, M. Barash, J. Chapman, On the algebraic properties of timeliness, Tech. rep., IOG (2023).
URL http://www.pnsol.com/public/Algebraic-Timeliness-TR.pdf

[16] K. S. Trivedi, Probability and Statistics with Reliability, Queuing, and Computer Science Applications, 2nd Edition, Wiley, New York, NY, USA, 2002.

[17] S. H. Haeri, P. W. Thompson, P. Van Roy, M. Haveraaen, N. J. Davies, M. Barash, K. Hammond, J. Chapman, Algebraic Reasoning About Timeliness, in: C. Aubert, C. Di Giusto, S. Fowler, L. Safina (Eds.), 16^{th} ICE, Vol. 383 of EPTCS, 2023, pp. 35–54. doi:10.4204/EPTCS.383.3.

[18] M. J. Gajda, Curious Properties of Latency Distributions, CoRR abs/2011.05219 (2020). doi:10.1007/978-3-031-10461-9_10.
URL https://arxiv.org/abs/2011.05219

[19] K. J. Sherry, Business Process Modelling with BPMN: Modelling and Designing Business Processes Course Book using The Business Process Model and Notation Specification Version 2.0, CreateSpace Independent Publishing Platform, 2012.

[20] P. Y. H. Wong, J. Gibbons, Formalisations and Applications of BPMN, SCP 76 (8) (2011) 633–650. doi:https://doi.org/10.1016/j.scico.2009.09.010.
URL https://www.sciencedirect.com/science/article/pii/S0167642309001282

[21] P. Y. H. Wong, J. Gibbons, Property Specifications for Workflow Modelling, SCP 76 (10) (2011) 942–967. doi:https:

//doi.org/10.1016/j.scico.2010.09.007.
URL https://www.sciencedirect.com/science/article/pii/S0167642310001735

[22] F. S. E. Ltd, Failures-divergence refinement: Fdr2 user manual (May 2012).
URL https://www.cs.ox.ac.uk/projects/concurrency-tools/download/fdr2manual-2.94.pdf

[23] O. El Hichami, M. Naoum, M. Al Achhab, I. Berrada, B. E. El Mohajir, An Algebraic Method for Analysing Control Flow of BPMN Models, iJES 3 (3) (2015) 20—26. doi:10.3991/ijes.v3i3.4862.
URL https://online-journals.org/index.php/i-jes/article/view/4862

[24] J.-P. Friedenstab, C. Janiesch, M. Matzner, O. Muller, Extending BPMN for Business Activity Monitoring, in: 45^{th} HICSS, 2012, pp. 4158–4167. doi:10.1109/HICSS.2012.276.

[25] C. Costello, O. Molloy, Towards a Semantic Framework for Business Activity Monitoring and Management, in: AAAI Spring Symposium: AI meets business rules and process management, 2008, pp. 17–27.

[26] L. E. M. Morales, Specifying BPMN Diagrams with Timed Automata: Proposal of Some Mapping Rules, in: 9^{th} CISTI, 2014, pp. 1–6. doi:10.1109/CISTI.2014.6876897.

[27] J. Hillston, A Compositional Approach to Performance Modelling, Cambridge University Press, 1996.

[28] R. De Nicola, D. Latella, A. L. Lafuente, M. Loreti, A. Margheri, M. Massink, A. Morichetta, R. Pugliese, F. Tiezzi, A. Vandin, The SCEL Language: Design, Implementation, Verification, Springer, 2015, pp. 3–71. doi:10.1007/978-3-319-16310-9_1.
URL https://doi.org/10.1007/978-3-319-16310-9_1

[29] L. Bortolussi, R. De Nicola, V. Galpin, S. Gilmore, J. Hillston, D. Latella, M. Loreti, M. Massink, CARMA: collective adaptive resource-sharing markovian agents, in: N. Bertrand, M. Tribastone (Eds.), Proc. 13^{th} W. Quant. Aspects of Prog. Lang. and Sys., Vol.

194 of EPTCS, 2015, pp. 16–31. doi:10.4204/EPTCS.194.2.
URL https://doi.org/10.4204/EPTCS.194.2

[30] W. Cazzola, F. Cesarini, L. Tansini, PerformERL: A Performance Testing Framework for Erlang, Distributed Comp. 35 (5) (2022) 439–454. doi:10.1007/s00446-022-00429-7.
URL https://doi.org/10.1007/s00446-022-00429-7

[31] Mil-std-1629a – procedures for performing a failure mode effect and criticality analysis, Tech. rep., United States Department of Defense (November 1980).

[32] K. Peachy, M. Race, V. Sri-Pathma, Post office scandal explained: What the horizon saga is all about (2024) [cited 17/1/2024].
URL https://www.bbc.com/news/business-56718036

[33] E. M. Stein, R. Shakarchi, Real Analysis: Measure Theory, Integration, and Hilbert Spaces, Princeton Uni. Press, 2005.

Declaration of interests

☒The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: