Towards Semantics-Aware Resilience in Composable Data Systems

Frédéric Loulergue[1][0000-0001-9301-7829] and Jolan Philippe[1][0000-0001-8759-4566]

Université d'Orléans, INSA Centre Val de Loire, LIFO UR 4022, Orléans, France
{frederic.loulergue,jolan.philippe1}@univ-orleans.fr

Abstract. Modern data management systems are increasingly assembled from specialized software components such as query optimizers, execution engines, and storage backends. They must also adapt to variations in available hardware resources, including accelerators, and to the availability of external analytical services. This composability is usually exploited for extensibility and performance. We argue that it should also be exploited for resilience. When resources become scarce, services disappear, or hardware capabilities change, a data system should be able to reconfigure itself and continue to deliver useful answers rather than fail abruptly. The difficulty is that reconfiguring a software component, changing an execution plan, or switching an execution backend may silently change the meaning of the computation. This paper presents a vision of semantics-aware resilience in composable data systems, in which a contract-guided resilience layer couples dynamic reconfiguration with semantic contracts. These contracts classify substitutions as exact, conditionally equivalent, or approximate with explicit bounds, and thereby turn resilience into controlled degradation rather than best-effort fallback. We discuss the main research challenges in contract design, multi-objective decision-making, benchmarking, and mechanically checkable substitution conditions.

Keywords: Composable data systems · dynamic reconfiguration · resilience · controlled degradation · formal semantics

1 Introduction

Digital infrastructures are increasingly exposed to disruptions that are not exceptional corner cases but recurring operational conditions: energy pressure, cooling limitations, partial hardware failures, network degradation, and service withdrawal. They must also cope with heterogeneous and sometimes ageing equipment. Resilience, understood as the ability to preserve useful service under adverse conditions, is therefore becoming a primary system's requirement rather than a secondary operational concern [25].

At the same time, the architecture of data management systems is changing. Instead of relying on monolithic database engines, modern platforms are

increasingly composed of specialized software components for storage management, query planning, and query execution [1,18]. They are also increasingly deployed on heterogeneous hardware, including accelerators. Systems and frameworks such as Apache Calcite [6], Velox [23], Apache Arrow DataFusion [19], BOSS [22], and CompoDB [14] show that composability has become a practical design principle in data management. This shift creates an opportunity: if multiple components or plans can realize the same analytical intent, then a system may reconfigure itself when its operating context changes [4,11].

However, resilience by reconfiguration is not just switching to a cheaper component. Replacing a GPU-accelerated operator with a CPU one, moving from full-data processing to sampling, or orchestrating several tools may affect semantics, precision, completeness, diversity, coverage, or cross-checking, not only latency and energy. Across several tools, orchestration choices may further affect the diversity and coverage of exploratory outputs, as well as the possibility of cross-checking them. In short, composability enables reconfiguration, but it also makes semantic drift easier. While prior work on component reconfiguration highlights the need for correctness arguments [10], and adaptive or robust query processing shows the value of runtime adaptation while preserving query semantics by construction [4,11,5,21], our concern is with explicit guarantees for degraded execution and cross-configuration substitution.

This paper presents a vision of *semantics-aware resilience* in composable data systems, where dynamic reconfiguration is guided by semantic contracts. These contracts allow a system to distinguish exact substitutions, substitutions valid only under explicit assumptions, and bounded quality degradation. Resilience then becomes controlled degradation rather than best-effort fallback. The novelty is not exact, conditional, and approximate substitutions taken separately, but their unification into a resilience-oriented contract layer for composable data systems.

The contributions of this vision paper are threefold. First, it frames resilience in composable data systems as a problem of semantic substitutability under changing resource and infrastructure constraints. Second, it proposes a contract-guided resilience layer for vertical reconfiguration in composable analytics stacks, discusses horizontal orchestration for exploratory analytics, and treats hybrid reconfiguration as the combination of these two dimensions. Third, it identifies the main research challenges raised by semantics-aware resilience, namely the design of semantic contracts, the design of resilient controllers, the construction of suitable benchmarks, and the mechanization of substitution reasoning.

The rest of the paper is organized as follows. Section 2 discusses the background and missing semantic layer. Section 3 introduces semantics-aware resilience and its reconfiguration scopes. Section 4 discusses the research agenda and a corresponding research path. We conclude in Section 5.

2 Background and Motivation

Composable Data Systems Change the Design Space. Composable data systems let engineers assemble platforms from reusable components rather than rebuild monolithic engines for each workload. Query planning can be delegated to Calcite [6], execution kernels can be reused through Velox [23], and modular analytics engines such as DataFusion [19] and BOSS [22] show that open, embeddable architectures can remain performant. The Cambridge Report on Database Research emphasizes research trends around modern hardware, sustainability, and heterogeneous application demands [1].

Yet the main benefits usually associated with composability remain performance, extensibility, and engineering reuse. The ability to *survive* degraded conditions is rarely treated as a first-class objective. When a service loses a GPU, exceeds an energy budget, experiences a cloud connector outage, or must run on older hardware, the common outcomes are performance collapse, emergency manual intervention, or outright service interruption. This is a poor match for many analytical settings where a slower, partial, or approximate answer may be far better than no answer at all.

From Failure Avoidance to Controlled Degradation. In constrained situations, continuity of service may matter as much as peak quality. For data analytics, this means preserving a useful service while adapting properties such as latency, update frequency, precision, or completeness. It may also require a temporary reduction in the set of available features. Such situations may be caused by climate-related events, supply-chain disruptions, geopolitical instability, or more routine operational incidents. Whatever the original cause, the data system eventually observes technical perturbations such as fewer nodes, a lower energy budget, missing accelerator resources, network instability, or the unavailability of an external software service.

This suggests a shift of perspective. We should not ask only whether the original configuration can still be executed. We should ask which alternative configuration can still satisfy the analytical intent under the new constraints, with the strongest guarantee that remains available. In this sense, resilience means controlled degradation rather than abrupt interruption. The system may have to move from exact processing to approximate processing, or from a full pipeline to a partial one. Such choices should be explicit, justified, and auditable.

The Missing Semantic Layer. Existing cost models, adaptive query processing [4,11], robust query processing [21,5], and resource schedulers can determine that one execution plan is cheaper than another, but they do not explain whether the substitute remains semantically valid. This limitation is particularly important in composable systems, where one may replace an operator implementation, change the data-consumption strategy, switch execution backends, or orchestrate several tools for the same analytical task. Such changes require more than performance comparisons:

– Some substitutions are *exact*: they preserve the intended result.

- Some are *conditionally equivalent*: they are correct only under explicit assumptions about the data, the workload, or the absence of nondeterminism.
- Some are *approximate*: they deliberately relax exactness but should provide explicit bounds or contracts on the deviation introduced.

What is missing, therefore, is a unified semantic layer to express and reason about these distinctions, which remain fragmented across the data stack. As a consequence, resilience choices remain ad hoc, hard to trust, and hard to compare.

3 A Vision of Semantics-Aware Resilience

3.1 A Contract-Guided Resilience Layer

Our vision is a data-system architecture in which runtime reconfiguration is governed not only by resource observations but also by semantic knowledge about candidate configurations. The starting point is an *analytical intent*: for example, compute an aggregate, execute a join-heavy pipeline, estimate a correlation, or carry out an interactive exploratory task. A configuration realizes this intent through a particular assembly of software components and a particular set of execution choices. These choices include parameter settings, data-access strategies, and hardware assignments.

We propose a *contract-guided resilience layer* between analytical intent and physical execution. It associates substitutable components, plans, and parameterization with semantic contracts. It observes perturbations such as memory pressure, missing accelerator resources, network instability, service unavailability, or explicit energy caps. It then selects and applies a substitute according to both resource constraints and semantic guarantees, and records the guarantee that is preserved. Under this view, adaptation is not merely a fallback mechanism. It is an explicit and traceable semantic choice. The proposed resilience layer could be realized within an autonomic management loop such as MAPE-K [17,26].

The primary target of this layer is *vertical reconfiguration*: changing components or parameters within one analytical stack while preserving the strongest guarantee available. A second dimension is *horizontal orchestration* across several tools or services for the same analytical task, especially in exploratory analytics. *Hybrid reconfiguration* combines these two dimensions. It covers cases where a system both reconfigures one analytical stack and orchestrates several tools or services within the same adaptive scenario.

3.2 Three Classes of Semantic Guarantees

The proposed contracts distinguish three classes of substitution. These classes describe the semantic relation between the original intent and the replacement configuration. A substitution may therefore replace one component by another component, one component by a composition of components, or one configuration by another configuration.

1. *Exact Substitutions.* These substitutions preserve the intended output. Typical examples include switching between two execution backends that implement the same operator semantics, or replacing one join algorithm with another under the same data model and query semantics. Exact substitutions support resilience without modifying user-visible meaning.
2. *Conditional Substitutions.* These substitutions preserve behaviour only under stated assumptions. For instance, two configurations may be equivalent only if the input satisfies a schema property, if duplicates are absent, or if ordering is irrelevant. Such substitutions are still valuable, but the assumptions must be represented rather than left implicit.
3. *Approximate Substitutions.* These substitutions intentionally trade quality for frugality or continuity. Examples include sampling, sketch-based aggregation, or reduced update frequency. Here the contract should provide a bounded notion of acceptable deviation: error on a numerical result, loss on a ranking metric, coverage reduction, or a more domain-specific measure of quality [15,16].

A simple vertical case illustrates the idea. Consider a service that computes `SUM(sales)` by region every five minutes for operational monitoring. Following a climate-related infrastructure incident, the service is moved from a primary site to a secondary site with fewer machines and no GPU accelerators. The resilience layer may then consider an exact CPU-based configuration that preserves the intended result but no longer meets the original refresh objective, or an approximate configuration based on sampling or sketches that preserves the refresh frequency with a declared error bound. A semantic contract makes this choice explicit: it records whether the substitute is exact or approximate, under which assumptions it is valid, and what degradation is introduced when the service continues under tighter resource constraints.

This example illustrates how the classification supports controlled degradation. The resilience layer can record whether the chosen substitute is exact, equivalent only under explicit assumptions, or approximate with a declared bound while respecting the new resource constraints. More generally, the classification provides a language for explaining and comparing resilience decisions.

3.3 Primary and Secondary Reconfiguration Scopes

Vertical Reconfiguration. In vertical reconfiguration, the system changes the internal composition of one analytical stack. This may involve replacing an operator implementation, changing a parameter, reassigning a computation between CPU and GPU, or activating or deactivating a module within the same pipeline. The analytical task remains the same, but the way it is carried out changes. This is the simplest setting in which to study semantics-aware resilience, because it allows one to compare several configurations for the same task within one stack.

Horizontal Orchestration. Polystore systems such as BigDAWG [12] show the practical interest of orchestrating heterogeneous engines. Different engines may

support different data models, execution strategies, or cost and quality trade-offs. Orchestration then makes it possible either to distribute parts of one analytical task across several engines or to compare several candidate results for the same task. In a resilience setting, the issue is then no longer only exact equivalence or bounded numerical error. It also concerns the diversity of the answers, the coverage of the explored search space, and the possibility of cross-checking one tool by another. If one service becomes unavailable, or if resource constraints force the system to reduce the number of active tools, the analytical service may continue with a reduced orchestration, but with reduced diversity, reduced coverage, or weaker cross-checking. In exploratory settings, one may also consider interestingness criteria such as novelty, relevance, surprise, or peculiarity [9]. This calls for contract notions different from those of vertical reconfiguration. Vertical contracts concern semantic preservation or bounded degradation of a task result, whereas horizontal contracts may concern evidence diversity, or source coverage.

Hybrid Reconfiguration. Hybrid reconfiguration combines vertical reconfiguration within one stack and horizontal orchestration across several tools or services. For example, an exploratory task may use multiple tools while one pipeline relies on sampling or approximate operators for a rapid first result. If confidence is too low or resources recover, selected parts of the analysis may then be refined through a more expensive configuration or through another tool. Hybrid reconfiguration is therefore relevant when resilience depends both on internal adaptation and on cross-tool coordination

4 Research Agenda

This vision raises four main research challenges.

4.1 Contracts for Analytical Intent and Substitutability

A first challenge is representational. What is the object that a contract talks about? One promising approach is to make the contract attach not merely to components in isolation but to well-defined parts of an analytical task: relational operators, distributed collection transformations, and pipeline fragments, with exploratory primitives treated as a later extension. The contract should state (i) the interface expected from the component or configuration; (ii) the assumptions under which the guarantee holds; (iii) the semantic guarantee provided; (iv) the resource envelope and observability requirements.

This raises a granularity trade-off. Contracts that are too coarse are easy to write but too weak for safe substitution. Contracts that are too fine become difficult to maintain and difficult for automated tooling to exploit. A central research question is therefore how rich a contract language must be to support useful resilience decisions without becoming unusable. In practice, such contracts may be introduced by different actors (e.g., system developers, library maintainers), with different versions. The resilience layer should treat these contracts as versioned artifacts, to avoid invalid substitution by introducing new semantics.

4.2 Decision-Making Under Multiple Objectives

A second challenge concerns decision-making under multiple objectives and semantic guarantees. Once several candidate substitutions are available, the controller must choose among them. The objective is not a single latency minimum but a compromise between resource consumption, analytical quality, and semantic status. Existing optimization techniques can help explore trade-offs between latency, energy, and quality. The difficulty here is that the controller must also account for whether a substitution is exact, valid only under explicit assumptions, or approximate with a declared bound. The precise optimization criteria may vary across scenarios: maintain exactness at all costs, preserve refresh frequency, or minimize energy under an acceptable error bound. In some settings, it may also be necessary to retain only a critical subset of analytical features.

This calls for a richer decision layer than a classical cost model. A resilient controller must reason over resource metrics, semantic guarantees, user-facing service policies, and the cost of reconfiguration itself. In other words, the decision problem is not *which plan is fastest?* but *which plan preserves the most valuable semantics under the current constraints?*

4.3 Benchmarks and Crisis Scenarios

A third challenge is empirical. A semantics-aware resilience benchmark must define perturbation scenarios, degradation policies, and metrics. Such metrics should include at least the preserved guarantee class, the observed quality loss and, when relevant, the error bound declared by the substitute, a service continuity measure such as latency or refresh frequency, and the cost of reconfiguration, for example the time needed to recover a stable service. Such a benchmark would be useful beyond this specific vision because it would make resilience properties measurable and comparable.

4.4 Mechanized Reasoning for Trustworthy Adaptation

A fourth challenge is formal. A large body of work exists on verification for component-based reconfiguration [10], and formal semantics have been mechanized for realistic fragments of SQL and Datalog [7,8]. However, the space of modern data-processing stacks remains too large to expect full mechanization in one step. Property-graph and modern query-language semantics illustrate the scale of the problem [2,13].

For this reason, we propose to start with Apache Spark as a practical platform for experimentation and as an initial target for formalization. Spark is expressive enough to cover both collection-style transformations and relational operations such as filtering, joins, grouping, and aggregation. It is also rich enough to express representative analytical pipelines while remaining more tractable than a full industrial query ecosystem. Previous work has already explored verified scalable parallel computing with Coq and Spark [20], as well as a Spark implementation based on a refined specification proved equivalent to an initial formal

specification [24]. Spark also remains a practical reference platform in large-scale analytics deployments [3]. A realistic first result is therefore a formalized kernel of Spark transformations together with mechanically checkable lemmas for exact vertical substitutions. Horizontal orchestration, hybrid reconfiguration, and richer forms of contract management can then be treated as later extensions.

4.5 Representative Cases

To make the vision concrete, we propose to start from a small set of representative cases. A first family of cases consists of exact vertical substitutions over representative analytical tasks such as filtering, aggregation, joins, and descriptive statistics. Examples include replacing execution backends, switching from GPU to CPU, or choosing between exact operator implementations with different cost profiles while showing that the resulting answers are unchanged.

A second family of cases consists of horizontal orchestration cases for exploratory analytics. Such cases involve several tools or services capable of addressing the same task, with attention to diversity, coverage, and cross-checking under resource constraints.

A third family covers hybrid cases that combine both dimensions. For example, several tools may be orchestrated to produce an initial result, while one pipeline already relies on sampling or approximate operators. If confidence is insufficient or resources recover, selected parts can later be refined using a more expensive configuration or another tool.

Another important object of study is the resilience layer itself. A prototype does not need to be a full new DBMS. It should demonstrate that perturbations can be mapped to contract-aware substitutions, that the preserved guarantee can be recorded, and that reconfiguration decisions can be explained in terms of both resource constraints and semantic guarantees.

Finally, some representative substitution patterns should be backed by *mechanically checkable reasoning*. A mechanized Spark core can serve as a reference semantics for selected operations, enabling proofs of equivalence for exact substitutions and proof obligations for conditional or approximate contracts.

5 Conclusion

In this paper, we presented a vision of semantics-aware resilience in composable data systems, where dynamic reconfiguration is guided by semantic contracts. These contracts classify substitutions as exact, conditional, or approximate with bounded quality loss. In this view, resilience becomes controlled degradation rather than best-effort fallback.

The contribution of this paper is to identify semantics-aware resilience as a data-management research problem in its own right. It requires work on specification, decision, benchmarking, and mechanized reasoning. It also suggests that composable architectures should be studied not only for performance and reuse,

but also for their ability to preserve a useful analytical service under degraded operating conditions.

As future work, we plan to design contract languages, define resilience benchmarks, develop representative prototypes, and mechanize representative substitution results. We will also study representative vertical cases, representative horizontal cases for exploratory analytics, and hybrid cases combining both dimensions, in order to assess the practical value and the semantic guarantees of the proposed approach. A natural first step is a Rocq formalization of core Spark transformations together with mechanically checkable lemmas for exact vertical substitutions. This would provide a first concrete basis on which to build.

Acknowledgements. Research on semantics-aware resilience in composable data systems is supported by the BQR programme of Université d'Orléans through the RADyD project, entitled "Resilience through dynamic adaptation for data processing".

References

1. Ailamaki, A., et al.: The Cambridge report on database research. CoRR **abs/2504.11259** (2025). https://doi.org/10.48550/ARXIV.2504.11259
2. Angles, R., et al.: PG-Schema: Schemas for property graphs. Proc. ACM Manag. Data **1**(2) (2023). https://doi.org/10.1145/3589778
3. Armbrust, M., et al.: Scaling Spark in the Real World: Performance and Usability. PVLDB **8**(12) (2015), http://www.vldb.org/pvldb/vol8/p1840-armbrust.pdf
4. Avnur, R., Hellerstein, J.M.: Eddies: Continuously adaptive query processing. In: International Conference on Management of Data (SIGMOD). ACM (2000). https://doi.org/10.1145/342009.335420
5. Babcock, B., Chaudhuri, S.: Towards a robust query optimizer: A principled and practical approach. In: International Conference on Management of Data (SIGMOD). ACM (2005). https://doi.org/10.1145/1066157.1066172
6. Begoli, E., Camacho-Rodríguez, J., Hyde, J., Mior, M.J., Lemire, D.: Apache Calcite: A foundational framework for optimized query processing over heterogeneous data sources. In: International Conference on Management of Data (SIGMOD). ACM (2018). https://doi.org/10.1145/3183713.3190662
7. Benzaken, V., Contejean, E.: A Coq mechanised formal semantics for realistic SQL queries: formally reconciling SQL and bag relational algebra. In: Certified Programs and Proofs (CPP). ACM (2019). https://doi.org/10.1145/3293880.3294107
8. Benzaken, V., Contejean, E., Dumbrava, S.: Certifying standard and stratified Datalog inference engines in SSReflect. In: Interactive Theorem Proving (ITP). LNCS, vol. 10499. Springer (2017). https://doi.org/10.1007/978-3-319-66107-0_12
9. Chanson, A., Labroche, N., Marcel, P., Peralta, V., Vassiliadis, P.: Interestingness measures for exploratory data analysis: a survey. In: ADBIS (Short Papers). Springer (2024). https://doi.org/10.1007/978-3-031-70421-5_2
10. Coullon, H., Henrio, L., Loulergue, F., Robillard, S.: Component-based distributed software reconfiguration: A verification-oriented survey. ACM Comput. Surv. **56**(1) (may 2024). https://doi.org/10.1145/3595376
11. Deshpande, A., Ives, Z.G., Raman, V.: Adaptive query processing. Found. Trends Databases **1**(1) (2007). https://doi.org/10.1561/1900000001

12. Duggan, J., Elmore, A.J., Stonebraker, M., Balazinska, M., Howe, B., Kepner, J., Madden, S., Maier, D., Mattson, T., Zdonik, S.B.: The BigDAWG polystore system. SIGMOD Rec. **44**(2) (2015). https://doi.org/10.1145/2814710.2814713
13. Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., Taylor, A.: Cypher: An evolving query language for property graphs. In: International Conference on Management of Data (SIGMOD). ACM (2018). https://doi.org/10.1145/3183713.3190657
14. Gavriilidis, H., Behme, L., Munz, C., Pandey, V., Markl, V.: CompoDB: A demonstration of modular data systems in practice. In: Simitsis, A., Kemme, B., Queralt, A., Romero, O., Jovanovic, P. (eds.) International Conference on Extending Database Technology, (EDBT). OpenProceedings.org (2025). https://doi.org/10.48786/EDBT.2025.97
15. Haas, P.J., Hellerstein, J.M.: Ripple joins for online aggregation. In: International Conference on Management of Data (SIGMOD). ACM Press (1999). https://doi.org/10.1145/304182.304208
16. Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online aggregation. In: International Conference on Management of Data (SIGMOD). ACM (1997). https://doi.org/10.1145/253260.253291
17. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1) (2003). https://doi.org/10.1109/MC.2003.1160055
18. Khurana, A., Le Dem, J.: The modern data architecture: The deconstructed database. login Usenix Mag. **43**(4) (2018), https://www.usenix.org/publications/login/winter-2018-vol-43-no-4/khurana
19. Lamb, A., Shen, Y., Heres, D., Chakraborty, J., Kabak, M.O., Hsieh, L.C., Sun, C.: Apache Arrow DataFusion: A fast, embeddable, modular analytic query engine. In: International Conference on Management of Data (SIGMOD). ACM (2024). https://doi.org/10.1145/3626246.3653368
20. Loulergue, F., Philippe, J.: Towards verified scalable parallel computing with Coq and Spark. In: Formal Techniques for Java-like Programs (FTfJP). ACM (2023). https://doi.org/10.1145/3605156.3606450
21. Markl, V., Raman, V., Simmen, D.E., Lohman, G.M., Pirahesh, H.: Robust query processing through progressive optimization. In: International Conference on Management of Data (SIGMOD). ACM (2004). https://doi.org/10.1145/1007568.1007642
22. Mohr-Daurat, H., Sun, X., Pirk, H.: BOSS - an architecture for database kernel composition. SIGMOD Rec. **54**(1) (Apr 2025). https://doi.org/10.1145/3733620.3733629
23. Pedreira, P., Erling, O., Basmanova, M., Wilfong, K., Sakka, L., Pai, K., He, W., Chattopadhyay, B.: Velox: Meta's unified execution engine. Proc. VLDB Endow. **15**(12) (2022). https://doi.org/10.14778/3554821.3554829
24. Philippe, J., Tisi, M., Coullon, H., Sunyé, G.: Executing certified model transformations on apache Spark. In: Software Language Engineering (SLE). ACM (2021). https://doi.org/10.1145/3486608.3486901
25. Welsh, T., Benkhelifa, E.: On resilience in cloud computing: A survey of techniques across the cloud domain. ACM Comput. Surv. **53**(3) (2020). https://doi.org/10.1145/3388922
26. White, S.R., Hanson, J.E., Whalley, I., Chess, D.M., Kephart, J.O.: An architectural approach to autonomic computing. In: International Conference on Autonomic Computing (ICAC). IEEE (2004). https://doi.org/10.1109/ICAC.2004.1301340