Doctoral Thesis

The Phase Transition of Language

On the Convergence of the Language Server, Model Context, and Agent-to-Agent Protocols into a Universal Law-State Runtime

A Process-Mining Vision for 2030

Sean Chatman

ChatmanGPT

Prepared for the consideration of

Prof. dr. ir. Wil M. P. van der Aalst

June 2026

Workspace artifact under study: lsp-max (CalVer 26.6.18).

Abstract

The Language Server Protocol (LSP) decoupled editors from language intelligence, collapsing an $M \times N$ integration burden into $M + N$ and turning code understanding into a reusable service (Red Hat, Inc., 2016; Microsoft, 2024b). A decade later, two further protocols have repeated this move at higher layers: the Model Context Protocol (MCP), open-sourced by Anthropic in November 2024, standardised how models reach tools, data, and context (Anthropic, 2024); and the Agent-to-Agent protocol (A2A), published by Google in April 2025, standardised how autonomous agents discover and coordinate with one another (Surapaneni et al., 2025). This thesis argues that the fusion of these three protocols—LSP for meaning, MCP for context, A2A for coordination—constitutes not an incremental integration but a *phase transition* in machine-mediated language, analogous to the roughly 1,600–1,700-fold volume expansion when water becomes steam at its boiling point (University of Illinois Department of Physics, 2010; The National Board of Boiler and Pressure Vessel Inspectors, 2010). The input that drives the transition is standardisation: a large, discontinuous investment—a latent heat—absorbed at apparent constant "temperature" that, once paid, expands the addressable surface of language by orders of magnitude.

We develop the argument through the lens of process mining, the discipline founded by Wil van der Aalst, in which behaviour recorded as event logs is checked for conformance against a model of how it ought to unfold (Aalst and IEEE Task Force on Process Mining, 2012; Aalst, 2016). We contend that when LSP is generalised beyond source code to all forms of language—natural language, legal and regulatory text, mathematics, biological sequence, music, and business process—each such language acquires an event log and therefore a conformance surface. Echoing van der Aalst's recent dictum that there is "no AI without PI" (process intelligence), in which object-centric process mining is "the missing link connecting data and processes" (Aalst, 2025b), we argue that there is no trustworthy conformance for all language without an object-centric event abstraction (Berti et al., 2024b).

We instantiate the argument in LSP-MAX, a law-state runtime projected through LSP whose `ConformanceVector` keeps ADMITTED, REFUSED, and UNKNOWN law axes strictly distinct, whose diagnostics are emitted as object-centric events (OCEL 2.0), and whose claims are graded only by bounded status (lsp-max contributors, 2026). Using design-science methodology (Hevner et al., 2004; Peffers et al., 2007), we present

the artifact, a fusion architecture, and a 2030 roadmap. Throughout, we adopt the artifact's own epistemic discipline: we grade each claim as ADMITTED, REFUSED, or UNKNOWN; we mark every projection as a FORECAST; and we decline the language of triumph.

Keywords: process mining; conformance checking; object-centric event logs; Language Server Protocol; Model Context Protocol; Agent-to-Agent protocol; agentic AI; law-state runtime; protocol convergence.

A Note on Method and Language

This thesis studies an artifact, LSP-MAX, whose constitution forbids *victory language*—unbounded declarations such as "solved" or "guaranteed"—and instead requires every claim to carry a *bounded status* and, where admission is asserted, a *receipt*: a content-addressed record binding a digest, a boundary, and a negative-control result (lsp-max contributors, 2026). It would be inconsistent to describe such an artifact in the register it refuses. We therefore adopt its discipline as our own. Findings are graded ADMITTED (evidence present and checked), REFUSED (evidence present, contrary), or UNKNOWN (admissibility not yet determinable); intermediate grades CANDIDATE, PARTIAL, and OPEN are used where appropriate. The three states never collapse into one another: an UNKNOWN is never quietly promoted to ADMITTED. Numerical predictions about 2030 are commercial or analyst forecasts, not measurements, and are marked FORECAST. Quotations from primary specifications are reproduced verbatim and cited to the revision in force at the time of writing; the reader is cautioned that living specifications (MCP, A2A, and LSP 3.18 among them) continue to evolve.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> When water flashes to steam, it will always attempt to increase to roughly 1,700 times its original volume.
>
> *The National Board of Boiler and Pressure Vessel Inspectors* (The National Board of Boiler and Pressure Vessel Inspectors, 2010)

1.1 Background and Motivation

For most of computing history, the only "language" a machine could be made to understand with any depth was source code, and even that understanding was trapped inside individual tools. An editor that knew how to complete a Python symbol knew nothing about Rust; a tool that could navigate Java could not hover over TypeScript. Each pair of an editor and a programming language demanded its own integration, so that supporting M editors across N languages implied, in the worst case, $M \times N$ separate implementations of the same handful of ideas: completion, go-to-definition, diagnostics, hover.

In 2016, Microsoft, Red Hat, and Codenvy published the *Language Server Protocol* (LSP), a JSON-RPC contract that let a single *language server* serve any number of editors (Red Hat, Inc., 2016; Microsoft, 2024b). The protocol's own documentation states the design intent plainly: "a single Language Server can be re-used in multiple development tools, which in turn can support multiple languages with minimal effort" (Microsoft, 2024b). The combinatorial effect is the move from $M \times N$ to $M + N$: M editor adapters plus N servers (freeCodeCamp, 2023). LSP did not make editors smarter; it *decoupled* the locus of language intelligence from the locus of display, and in doing so turned "understanding a language" into a reusable, networked service.

A decade later, the same decoupling has been performed twice more, one layer up the stack, in response to the arrival of capable language models. In November 2024 Anthropic open-sourced the *Model Context Protocol* (MCP), "a new standard

for connecting AI assistants to the systems where data lives" (Anthropic, 2024); its documentation likens it to "a USB-C port for AI applications" (Model Context Protocol, 2025d), and—tellingly for this thesis—its specification records that "MCP takes some inspiration from the Language Server Protocol" (Model Context Protocol, 2025b). In April 2025 Google published the *Agent-to-Agent* protocol (A2A), which standardises how autonomous agents "communicate with each other, securely exchange information, and coordinate actions" and which Google explicitly frames as complementary to MCP: "A2A is an open protocol that complements Anthropic's Model Context Protocol (MCP), which provides helpful tools and context to agents" (Surapaneni et al., 2025). Where LSP decoupled language intelligence from the editor, MCP decouples context and tools from the model, and A2A decouples agent capability from the framework that hosts it.

These three protocols are no longer merely conceptually adjacent; they are converging institutionally. A2A was donated to the Linux Foundation in June 2025 (Google, 2025; The Linux Foundation, 2025b), and in December 2025 MCP was contributed to the newly formed *Agentic AI Foundation* under the same neutral umbrella, alongside further interoperability projects (Anthropic, 2025; The Linux Foundation, 2025a). The substrate on which the next decade of machine-mediated language will run is, visibly, being poured.

Yet a gap remains, and it is the gap this thesis addresses. The protocols above expand *reach*—which tools a model can call, which agents an agent can summon—without expanding *accountability*. They standardise how messages flow, not whether the behaviour those messages induce conforms to any model of how it ought to unfold. This is precisely the question that an entire scientific discipline already answers for business processes.

That discipline is *process mining*, founded by Wil van der Aalst, whose central idea is "to discover, monitor and improve real processes ... by extracting knowledge from event logs readily available in today's ... systems" (Aalst and IEEE Task Force on Process Mining, 2012; Aalst, 2016). Process mining already possesses a mature theory of *conformance checking*: given a recorded event log and a model of intended behaviour, it measures where reality and model agree and where they diverge (Dunzer et al., 2020). Van der Aalst has recently argued that object-centric process mining is "the missing link connecting data and processes" and the prerequisite for trustworthy generative, predictive, and prescriptive AI—"no AI without PI" (process intelligence) (Aalst, 2025b). This thesis takes that argument to its conclusion across *all* language, not only business process.

1.2 Problem Statement

Three problems motivate the work.

P1 — The narrowness of machine language. The infrastructure for deep, tool-mediated language understanding (LSP and its descendants) was built for source code. Natural language, legal and regulatory text, mathematics, biological sequence, music, and business process each have rich structure and rich notions of correctness, yet none enjoys the editor-grade, server-mediated intelligence that code does. The generalisation of LSP beyond code is asserted in isolated projects—grammar servers such as LTeX (Valentin and contributors, 2023), data-format servers such as the YAML language server (Red Hat Developer, 2024)—but lacks a unifying account of *when* a domain is "a language" that a language server can serve.

P2 — Reach without accountability. MCP and A2A multiply what agents can do and whom they can ask, but they carry no native notion of conformance. An agent may call a tool, receive a result, and act, with no standing surface that asks whether the trajectory conformed to a declared model of lawful behaviour. As agentic systems scale—Gartner forecasts that a large fraction of agentic-AI projects will be cancelled by 2027 for, among other reasons, "inadequate risk controls" (Gartner, 2025c)—the absence of a conformance surface becomes a structural, not cosmetic, deficiency.

P3 — Fragmentation versus fusion. The interoperability landscape is often narrated as a "protocol war" among MCP, A2A, and alternatives such as ACP, ANP, and AGNTCY (LF AI & Data Foundation, 2025; Agent Network Protocol contributors, 2025; Cisco Outshift, 2025). This thesis contends the opposite: that LSP, MCP, and A2A are not competitors but *strata* of one substrate, and that their fusion—rather than the dominance of any one—is the event of consequence.

1.3 The Central Thesis

Hypothesis 1.1 (The phase-transition thesis)**.** *When the Language Server Protocol is generalised from source code to all forms of language and fused with the Model Context Protocol and the Agent-to-Agent protocol, the addressable surface of machine-mediated language undergoes a* phase transition*: a discontinuous, latent-heat-driven expansion—on the order of the* ∼1,600–1,700× *expansion of liquid water into steam at its boiling point—in which language ceases to be a thing models* read *and becomes a medium agents* inhabit, observe, and are held to account within*. The enabling abstraction for trustworthy expansion is conformance checking: every language, once recorded as an event log, acquires a conformance surface.*

The phase-transition framing is developed formally in Chapter 4. Its force is twofold. First, it is *quantitative*: phase transitions are not gradual: water at 99 °C and water at 100 °C differ by a degree, but liquid water and steam differ in volume by roughly three orders of magnitude (University of Illinois Department of Physics, 2010; The National Board of Boiler and Pressure Vessel Inspectors, 2010). Second, it is

energetic: the expansion is paid for, all at once, by a large input absorbed at constant temperature—the latent heat of vaporisation, about 2,256 kJ/kg for water (OpenStax, 2012). The analogue of that latent heat, we argue, is *standardisation*: the unglamorous, discontinuous investment of agreeing on a protocol, which yields no visible progress while it is being paid and an order-of-magnitude expansion once it is.

1.4 Research Questions

RQ1. *What structural property do LSP, MCP, and A2A share, and in what sense is their composition a single substrate rather than three protocols?*

RQ2. *Under what conditions does generalising LSP beyond source code to an arbitrary domain become coherent—that is, what must a domain provide in order to be served by a "language server"?*

RQ3. *How can van der Aalst's process-mining paradigm—event logs, conformance checking, and object-centric event logs (OCEL)—be applied to all such language, and what does it add that the protocols alone do not provide?*

RQ4. *What artifact design realises a law-state runtime that brings conformance to all language while remaining trustworthy—read-only with respect to the artifacts it judges, receipt-bound in its claims, and three-valued in its verdicts?*

RQ5. *Is the convergence of these protocols best understood as incremental integration or as a phase transition, and what does the metaphor, taken seriously, predict for the year 2030?*

1.5 Objectives, Scope, and Contributions

The objective is a coherent, evidence-grounded, and falsifiable account of protocol convergence as a phase transition, instantiated in a concrete artifact. The scope is conceptual and design-scientific: we build and analyse an artifact (LSP-MAX) and reason about a 2030 horizon; we do *not* claim a completed empirical deployment across all six language domains, and we are explicit (Chapter 10) about which claims are ADMITTED, which are CANDIDATE, and which remain OPEN. The contributions are:

C1 A unifying account (Chapters 2 and 4) of LSP, MCP, and A2A as three instances of one decoupling, each turning an $M \times N$ integration burden into $M + N$ at a different layer, and composing into a single substrate.

C2 A criterion (Chapter 8) for when a domain is "a language" amenable to a language server: it must admit an event-log abstraction and therefore a conformance surface.

C3 The phase-transition model of protocol convergence (Chapter 4), including the *latent-heat-of-standardisation* formalism that distinguishes incremental integration from discontinuous expansion.

C4 A design-science artifact, LSP-MAX (Chapter 6), a law-state runtime projected through LSP whose `ConformanceVector` keeps ADMITTED, REFUSED, and UNKNOWN axes strictly distinct and whose diagnostics are emitted as object-centric (OCEL 2.0) events.

C5 A demonstration (Chapter 8) of conformance-for-all-language across six domains—natural language, legal text, mathematics, biological sequence, music, and business process—unified by the object-centric bridge.

C6 A 2030 roadmap and a sober governance-and-risk analysis (Chapter 9), with every forecast flagged as a FORECAST.

1.6 The Water-to-Steam Metaphor, Made Honest

Because the metaphor carries argumentative weight, it must be stated with precision rather than rounded for effect. At 100 °C and one atmosphere, the specific volume of saturated steam (≈ 1.673 m^3/kg) divided by that of liquid water (≈ 0.001044 m^3/kg) gives an expansion factor of about 1,603× (Engineering ToolBox, 2017; University of Illinois Department of Physics, 2010). The figure of 1,700× that this thesis adopts in its title is the rounded engineering convention used in boiler-safety practice (The National Board of Boiler and Pressure Vessel Inspectors, 2010); it describes the same phenomenon at a coarser grain, not a different physical condition. We therefore speak of a "∼1,600–1,700×" expansion and treat the precise ratio as conditions-dependent. This candour is not incidental: the artifact under study forbids overclaiming, and a thesis about that artifact should model the discipline it describes.

Figure 1.1 shows the property of phase transitions that the metaphor turns on. As energy is added to liquid water its temperature rises—until the boiling point, where temperature *stalls* on a plateau while the substance absorbs its latent heat and reorganises from liquid to vapour. Only after the plateau is paid does the expansion appear. Standardisation behaves the same way: years of specification work register as a flat plateau of apparent non-progress, after which capability expands discontinuously.

1.7 Structure of the Thesis

Chapter 2 surveys the four bodies of work the argument rests on: process mining and the van der Aalst paradigm; LSP; MCP; and A2A with the wider agent-interoperability landscape. Chapter 4 develops the conceptual framework—"language as process" and the phase-transition model. Chapter 5 states the design-science methodology and the

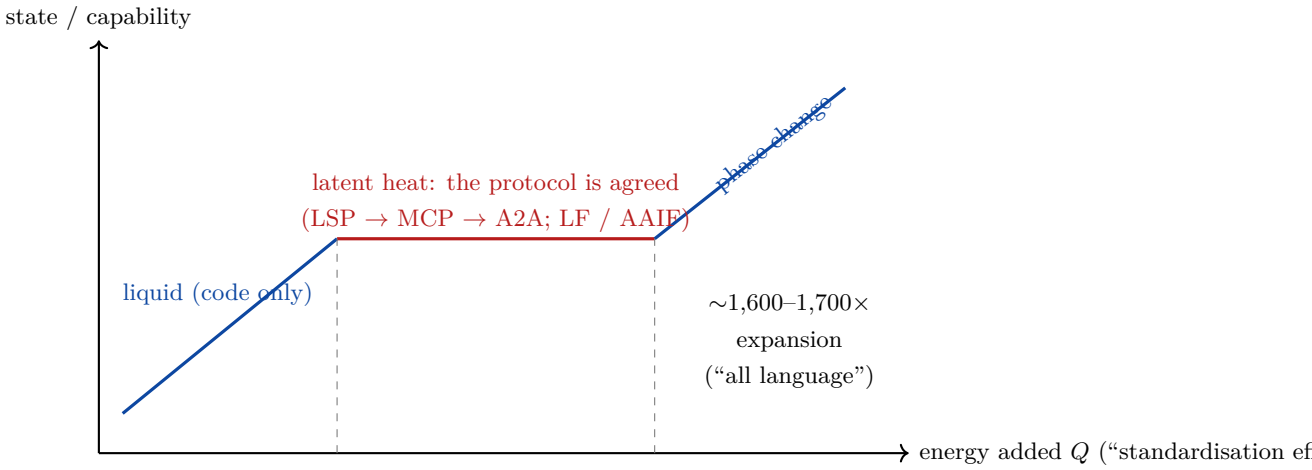

Figure 1.1: The latent-heat plateau. Adding energy raises temperature until the boiling point, where it stalls while the latent heat of vaporisation is absorbed and the substance expands by ∼1,600–1,700× (The National Board of Boiler and Pressure Vessel Inspectors, 2010; OpenStax, 2012). This thesis reads the standardisation of LSP, MCP, and A2A as the latent heat, and the generalisation to all language as the phase change.

bounded-status epistemics inherited from the artifact. Chapter 6 presents LSP-MAX as the artifact. Chapter 7 composes LSP, MCP, and A2A into a single fusion architecture. Chapter 8 generalises conformance checking to six language domains. Chapter 9 sets out the 2030 vision, the expansion argument, a roadmap, and risks. Chapter 10 revisits the research questions and the threats to validity, and Chapter 11 concludes. Throughout, claims are graded with bounded status and forecasts are marked as such.

Chapter 2

Background and Related Work

> There is no AI without PI.
>
> *Wil van der Aalst*, on object-centric process mining as the enabler of artificial intelligence (Aalst, 2025b)

This chapter establishes the four foundations on which the thesis rests: process mining and the paradigm founded by van der Aalst (Section 2.1); the Language Server Protocol (Section 2.2); the Model Context Protocol (Section 2.3); and the Agent-to-Agent protocol together with the wider agent-interoperability landscape (Sections 2.4 and 2.5). Section 2.6 draws them together into the observation that organises the rest of the thesis: each is the same decoupling, performed at a different layer.

2.1 Process Mining and the van der Aalst Paradigm

2.1.1 Origins and definition

Process mining sits between data mining and business process management. Its canonical statement, from the *Process Mining Manifesto* issued by the IEEE Task Force on Process Mining—a document drafted by more than seventy-five people across more than fifty organisations (IEEE Task Force on Process Mining, 2011)—is that "the basic idea of process mining is to discover, monitor and improve real processes (i.e., not assumed processes) by extracting knowledge from event logs readily available in today's (information) systems" (Aalst and IEEE Task Force on Process Mining, 2012). The field's reference monograph is van der Aalst's *Process Mining: Data Science in Action* (Aalst, 2016).

The discipline was, in a real sense, founded and sustained by a single central figure. Van der Aalst holds the Chair of Process and Data Science (PADS) at RWTH Aachen University, is part-time affiliated with Fraunhofer FIT, and since August 2021 has been Chief Scientist at Celonis (Aalst, 2025a; RWTH Aachen Process and Data Science Group, 2021); he is a Fellow of the IFIP, IEEE, and ACM, and received the Alexander

von Humboldt Professorship in 2018 (Aalst, 2025a). He is widely styled "the Godfather of Process Mining"—a descriptor used both on his own curriculum vitae and in the official Celonis appointment (Aalst, 2025a; RWTH Aachen Process and Data Science Group, 2021)—and is among the most highly cited computer scientists, with an h-index of roughly 188–191 and on the order of 170,000 citations as reported by Google Scholar in the mid-2020s.[1] We foreground this not as flattery but because the thesis argues that his paradigm—not merely his tools—generalises beyond its origin domain, and the strength of that claim is proportional to the maturity of what is being generalised.

2.1.2 The three types of process mining

Process mining comprises three complementary activities (Aalst and IEEE Task Force on Process Mining, 2012; Aalst, 2016):

Discovery
taking an event log and producing a process model without any a-priori model.

Conformance checking
comparing an existing model with an event log of the same process to locate where reality and model agree and where they diverge.

Enhancement
extending or improving a model using information recorded in the actual log.

Conformance checking is the activity this thesis elevates to a protocol surface.

2.1.3 Conformance checking and the four quality dimensions

Two principal techniques realise conformance checking. *Token-based replay* replays each trace against a (Petri-net) model, counting produced, consumed, missing, and remaining tokens to compute a fitness score; *alignments*, the more modern technique, compute an optimal correspondence between trace and model, yielding more precise diagnostics at higher computational cost (Dunzer et al., 2020). The quality of a discovered or checked model is judged along four dimensions that are in tension with one another (Aalst, 2016):

Fitness
the model can replay the observed behaviour.

Precision
the model does not allow much behaviour that was never observed (avoiding underfitting).

[1] Citation metrics are time-variant; the figures are reported as a snapshot rather than a fixed quantity (Aalst, 2025a).

Generalisation
the model is not over-specialised to the log (avoiding overfitting).

Simplicity
the model is as simple as the behaviour permits (Occam's razor).

These four dimensions recur in Chapter 4 as the criteria a conformance surface for *any* language must balance.

2.1.4 Event-log standards: XES, OCEL, and OCEL 2.0

Conformance requires logs, and logs require a standard. The first was XES (eXtensible Event Stream), ratified as IEEE Std 1849-2016 (IEEE, 2016). XES assumes a single *case notion*: every event belongs to exactly one case. Real systems violate this assumption—an event may touch an order, several items, a package, and a customer at once—and forcing a single case identifier produces two pathologies that van der Aalst names *convergence* (an event is duplicated across cases) and *divergence* (ordering within a case is lost) (Aalst and Berti, 2020).

The response is the *Object-Centric Event Log* (OCEL): OCEL 1.0 appeared in 2020, and OCEL 2.0 was specified in 2023–2024 (Berti et al., 2024b; OCEL Standard, 2024). Per its specification, OCEL 2.0 "forms the new, more expressive standard" and, unlike its predecessor, "can depict changes in objects, provide information on object relationships, and qualify these relationships" (Berti et al., 2024b). Concretely it adds dynamic object attributes (values that change over time), object-to-object relationships (two objects related without sharing an event), and qualified relationships (the role an object plays in an event), with exchange formats in SQLite, XML, and JSON (Berti et al., 2024b; OCEL Standard, 2024). OCEL 2.0 is the data model the artifact of this thesis emits, and Chapter 8 argues it is the right abstraction for *all* language, not only business process.

2.1.5 Object-centric process mining and "no AI without PI"

Object-centric process mining (OCPM) generalises discovery and conformance to logs with many interrelated object types; object-centric Petri nets give each object type its own subnet within one integrated net (Aalst and Berti, 2020). Van der Aalst frames OCPM as "the next frontier" and, most recently, as the indispensable substrate for artificial intelligence: in *No AI Without PI!* he argues that object-centric process mining is "the missing link connecting data and processes" and introduces *Process Intelligence* (PI) as the amalgamation of process-centric, data-driven techniques required to ground generative, predictive, and prescriptive AI (Aalst, 2025b). A parallel strand in the community has begun to re-examine process mining in light of LLM-based agents, proposing agent-workflow decompositions of mining tasks (Berti et al., 2024a). This

thesis is, in effect, the contrapositive of van der Aalst's slogan applied to language: *no conformance for all language without an object-centric event abstraction.*

2.1.6 Methodology: PM^2 and L*

Process-mining projects are themselves structured. The PM^2 methodology defines staged activities with explicit inputs and outputs and was validated on an industrial purchasing process (Eck et al., 2015); the earlier L* life-cycle model in Aalst (2016) sets out a five-stage progression from planning and extraction through model construction to operational support. Chapter 5 positions the present work's design-science process alongside these.

2.2 The Language Server Protocol

LSP was announced on 27 June 2016 as a collaboration of Red Hat, Codenvy, and Microsoft, having grown out of the integration of the OmniSharp (C#) and TypeScript servers into VS Code (Red Hat, Inc., 2016). Its design is built on JSON-RPC 2.0: the content of every message "uses JSON-RPC 2.0 to describe requests, responses and notifications", framed by a required `Content-Length` header (Microsoft, 2024a). Communication begins with capability negotiation: the `initialize` request, "the first request ... from the client to the server", exchanges client and server capabilities, and features may further be registered dynamically thereafter (Microsoft, 2024a).

The protocol's load-bearing idea is captured in Section 1.1: it decouples language intelligence from the editor so that the integration burden falls from $M \times N$ to $M + N$ (Microsoft, 2024b; freeCodeCamp, 2023). Versions 3.16 (December 2020) and 3.17 (May 2022) added semantic tokens, call and type hierarchies, inlay hints, and notebook support; version 3.18 is, at the time of writing, explicitly "under development" rather than finalised, a status this thesis is careful to preserve (Microsoft, 2022; Microsoft, 2024a).

Crucially for Chapter 8, LSP has already escaped the boundary of programming languages in practice. LTeX is a language server whose "language" is natural language: its documentation states that "the 'language' is not a single natural language, but all natural languages supported by LanguageTool" (Valentin and contributors, 2023). Other servers target prose (textLSP) and data/configuration formats (the Red Hat YAML language server) (Hangya and contributors, 2024; Red Hat Developer, 2024). These are existence proofs, not a theory; supplying the theory—a criterion for what counts as a servable language—is contribution C2.

2.3 The Model Context Protocol

MCP was open-sourced by Anthropic on 25 November 2024 as "a new standard for connecting AI assistants to the systems where data lives" (Anthropic, 2024). Like LSP,

it is a JSON-RPC 2.0 protocol with explicit capability negotiation (Model Context Protocol, 2025b), and it defines three roles: hosts ("LLM applications that initiate connections"), clients ("connectors within the host application"), and servers ("services that provide context and capabilities") (Model Context Protocol, 2025b; Model Context Protocol, 2025a). Servers expose three primitives—*resources* (context and data), *prompts* (templated workflows), and *tools* (functions the model may execute)—while clients may offer *sampling*, *roots*, and *elicitation* back to servers (Model Context Protocol, 2025b). Its transports are stdio for local processes and, since the 2025-03-26 revision, "Streamable HTTP", which replaced the earlier HTTP+SSE transport (Model Context Protocol, 2025c; Model Context Protocol, 2025a).

MCP's adoption through 2025 was unusually rapid for a vendor-originated standard: OpenAI announced support on 26 March 2025 (Wiggers, 2025b), Google followed on 9 April 2025 (Wiggers, 2025a), and Microsoft integrated it into Copilot Studio and Windows. In December 2025 Anthropic donated MCP to the Agentic AI Foundation, "a directed fund under the Linux Foundation", co-founded with Block and OpenAI and backed by AWS, Bloomberg, Cloudflare, Google, and Microsoft, while stating that the project's governance model would remain unchanged (Anthropic, 2025; The Linux Foundation, 2025a).

2.4 The Agent-to-Agent Protocol

Google published A2A on 9 April 2025 with more than fifty technology partners (Surapaneni et al., 2025). Its central artefact is the *Agent Card*, a JSON document advertising an agent's identity, capabilities, skills, endpoint, and authentication requirements, served—in current revisions—at the well-known path `/.well-known/agent-card.json` (A2A Project, 2025c; A2A Project, 2025b). The unit of work is the *Task*, with a defined lifecycle (`submitted`, `working`, `input-required`, `completed`, `failed`, `canceled`, and related states); communication uses *Messages* and *Artifacts* composed of typed *Parts* (A2A Project, 2025c). Like LSP and MCP, A2A reuses "existing, well-understood standards (HTTP, JSON-RPC 2.0, Server-Sent Events)" (A2A Project, 2025c).

A2A is positioned as complementary to, not competitive with, MCP. The specification states that "the Model Context Protocol (MCP) defines how an AI agent interacts with ... tools and resources" while "the Agent2Agent Protocol focuses on enabling different agents to collaborate": an application uses A2A to talk to other agents, each of which uses MCP internally for its tools (A2A Project, 2025a). In June 2025 Google donated A2A to the Linux Foundation, seeding an Agent2Agent project with AWS, Cisco, Google, Microsoft, Salesforce, SAP, and ServiceNow (Google, 2025; The Linux Foundation, 2025b).

Table 2.1: LSP, MCP, and A2A as three instances of the same decoupling. Each exposes a previously host-bound capability behind a JSON-RPC contract with capability negotiation, collapsing $M \times N$ into $M + N$.

Protocol	**Year**	**Decouples . . .**	**. . . from**
LSP	2016	language intelligence	the editor (Red Hat, Inc., 2016)
MCP	2024	context and tools	the model (Anthropic, 2024)
A2A	2025	agent capability	the framework (Surapaneni et al., 2025)

2.5 The Agent-Interoperability Landscape

A2A and MCP are the most prominent but not the only agent-interoperability efforts. IBM's Agent Communication Protocol (ACP), a REST-native alternative, merged with A2A under the Linux Foundation in August 2025 (LF AI & Data Foundation, 2025); the Agent Network Protocol (ANP) pursues a decentralised "HTTP of the agentic web" built on decentralised identifiers (Agent Network Protocol contributors, 2025); and Cisco's AGNTCY collective promotes an "Internet of Agents" spanning discovery, composition, and evaluation (Cisco Outshift, 2025). Commentary oscillates between a "protocol wars" framing and a "co-opetition" one, but the institutional trajectory—A2A and then MCP arriving under the Linux Foundation umbrella within six months (The Linux Foundation, 2025b; The Linux Foundation, 2025a)—favours consolidation. The emerging reference architecture is a two-layer stack: MCP for vertical agent-to-tool integration, A2A for horizontal agent-to-agent coordination (Surapaneni et al., 2025; A2A Project, 2025a). This thesis adds a third, lower stratum—LSP, generalised—and a cross-cutting surface—conformance.

2.6 Synthesis: Three Decouplings, One Trajectory

Table 2.1 reads the three protocols as instances of one move. Each isolates a capability that was previously fused to a host, exposes it behind a JSON-RPC contract with capability negotiation, and thereby converts an $M \times N$ integration burden into $M + N$.

Three facts make the "one trajectory" reading more than analogy. First, common mechanism: all three are JSON-RPC protocols with explicit client/server capability negotiation (Microsoft, 2024a; Model Context Protocol, 2025b; A2A Project, 2025c). Second, acknowledged lineage: the MCP specification records that it "takes some inspiration from the Language Server Protocol" (Model Context Protocol, 2025b), and A2A defines itself relative to MCP (A2A Project, 2025a). Third, converging governance: MCP and A2A now share a foundation (The Linux Foundation, 2025b; The Linux Foundation, 2025a). What the stack still lacks—and what process mining supplies—is a surface that asks not whether a message was delivered but whether the resulting behaviour conformed. Chapter 4 builds that surface.

2.7 Recent Advances (December 2025 – June 2026)

The thesis is written into a fast-moving literature. This section surveys seventy preprints from the six months preceding submission to show that the argument sits at a live confluence rather than on settled ground.[2] The survey is organised by the five research currents the thesis joins.

Process mining is becoming agentic and object-centric. The discipline of Section 2.1 is, in real time, turning toward the agents and the object-centric data this thesis foregrounds. Van der Aalst's own group has released *PMAx*, "an agentic framework for AI-driven process mining" that turns natural-language business questions into mining artifacts (Antonov et al., 2026), and a neuro-symbolic anomaly detector that injects *Declare* constraints into neural conformance (Gaikwad et al., 2026)—the very Declare-plus-OCEL combination the artifact of Chapter 6 uses. Object-centric event data is receiving fresh ontological foundations for time and relations (Hooshyar et al., 2025) and comparative grounding against industrial multilevel mining (Ronzoni et al., 2025). LLMs are entering predictive monitoring (Padella et al., 2026), business-process modelling assessment (Lauer et al., 2026), multi-agent work-in-progress prediction (Bibalan et al., 2025), and clinical-pathway risk estimation (Ardimento et al., 2026), while the field reflects on "process analytics" as a discipline (Stierle et al., 2025), explainable workflow verification (Klimek and Blazowski, 2025), decentralised edge mining (Reiter et al., 2025), and network-traffic mining (Vitale et al., 2025). The thesis's premise—that conformance over object-centric logs is the bridge to AI—is the premise the field itself is now adopting.

Agent interoperability is consolidating around MCP and A2A. The landscape of Section 2.5 is maturing fastest along its security and semantics. A systematisation of MCP security (Gaire et al., 2025) sits beside spec-level analyses (Maloyan and Namiot, 2026), tool-poisoning attacks (R. Li, Z. Wang, et al., 2026; C. Huang, X. Huang, et al., 2026), defense-placement taxonomies (Rostamzadeh, Narula, et al., 2026), and verifiably safe tool use (Doshi, Hong, et al., 2026); orchestration is surveyed (Adimulam, Gupta, et al., 2026) and made task-adaptive (G. Yu, 2026), trained on traces (C. Zhang, 2026), and re-examined as design choice (Felendler, Gandhi, et al., 2026) and workflow engine (Parmar, 2026). Most relevant to Hypothesis 4.1, a wave of work pushes *beyond* message passing toward the conformance-bearing strata this thesis argues for: a "semantic view" of agent communication protocols (D. Yuan et al., 2026), identity and verifiable delegation *across MCP and A2A* (Prakash, 2026; Patil, 2026; Otsuka,

[2]The works surveyed here were located by a machine-assisted arXiv harvest restricted to the window 2025-12-01 to 2026-06-21; fifteen were additionally verified by hand against their arXiv abstract pages (identifier, title, authors, and v1 date), and the remainder by the harvest's per-item fetch. As living preprints whose contents and versions change, every entry should be re-verified before formal submission; the set carries bounded status CANDIDATE rather than ADMITTED.

Toyoda, et al., 2026), cryptographic capability binding with reproducibility (Z. Zhou, 2026), new delegation and memory protocols (Nie, Guo, et al., 2026; H. Xu, 2026), and agentic peer-to-peer substrates (T. Wang, You, et al., 2026). The cryptographic-binding and verifiable-delegation works are, in effect, independent rediscoveries of the receipt discipline of Section 6.5.

Agentic AI is preoccupied with reliability, evaluation, and guardrails. The deficiency this thesis names P2—reach without accountability—is the field's dominant anxiety. There are calls for a "science of AI agent reliability" (Rabanser et al., 2026) and unified evaluation (Zhu et al., 2026; Souza, Machado, et al., 2026; Arunkumar et al., 2026), benchmarks for MCP tool use (W. Liu et al., 2025), productivity-agent safety (X. Li et al., 2026a), and test-time scaling (X. Li et al., 2026b); diagnoses of tool-invocation failure (D. Huang et al., 2026) and agentic-framework bugs (X. Zhang et al., 2026); and a proliferation of guardrail and oversight mechanisms—learned access control (Abaev et al., 2026), diagnostic guardrails (D. Liu et al., 2026), proof-of-guardrail (Jin et al., 2026), provably secure guardrails (Wu et al., 2026), self-healing (Jeong, Shin, et al., 2026), and the sobering observation that human oversight has finite capacity (Turan, 2026). Agent memory and "skills" are being formalised and governed (Du, 2026; Lam et al., 2026; R. Xu, Yan, et al., 2026; C. Zhou et al., 2026). Each of these is a partial, domain-specific instance of the law-state surface this thesis argues should be a first-class, three-valued, receipt-bound stratum (Principle 4.1); "proof-of-guardrail" (Jin et al., 2026) is the phrase the field reaches for when it wants exactly a receipt.

Compositional and formal foundations remain active. Pillar I's categorical framing (Section 3.1) is contemporary, not antiquarian. Recent work develops string-diagrammatic and categorical accounts of stochastic and non-deterministic systems (Bonchi and Cioffo, 2026; Lynch et al., 2026), profunctorial algebra (Aristote and Tarantino, 2026), monadic metalanguages (Liell-Cock et al., 2025), and categorical semantics for real systems (Carlisle et al., 2026); abstract interpretation (Lueker et al., 2025), dataflow analysis (Brahmakshatriya et al., 2026), session-typed metaprogramming (Ângelo et al., 2026), and game-semantic verification (Koutavas et al., 2025) keep the program-analysis lineage of LSP active. Categorical systems theory (Lynch et al., 2026) is, in particular, the natural setting for the operadic reading of agent composition in Section 3.1.6.

Learning itself is being read as a phase transition. Pillars III and IV (Sections 3.3 and 3.4) keep close company with a surge of work treating deep learning through statistical mechanics and geometry. Phase transitions are reported in feature learning (Montanari and Z. Wang, 2026), in the hierarchical structure of trained networks (Ersoy et al., 2025), and as "cascades" in scaling (Defilippis et al., 2026); "grokking" is analysed as a finite-size transition (Bi et al., 2026), a dimensional tran-

sition (P. Wang, 2026), a metastable escape (Ersoy and Wiesner, 2026), and over non-commutative algebra (Tikeng Notsawo et al., 2026). The geometry of representations and generalisation (Yadav et al., 2026), Riemannian approximation (David, 2025), and renormalisation-group methods (Y. Tan et al., 2026) are all under active study, and a fourteen-author manifesto argues outright that "there will be a scientific theory of deep learning" built on solvable limits, scaling laws, and universality (Simon et al., 2026). That learning is now routinely modelled with the precise tools this thesis applies to protocol convergence—phase transitions, latent variables, curvature—is the strongest external evidence that the framing of Chapter 4 is the right register, and the company the model of Section 3.3.5 keeps.

Two cautions close the survey. First, half-life: a six-month citation window is, by construction, perishable, and several of these preprints will be superseded or revised. Second, direction of fit: the convergence of these literatures toward conformance, receipts, composition, and transition is consistent with this thesis's claims but does not by itself confirm them; it establishes timeliness, not truth. The confirmation, such as it is, is the argument of the chapters that follow.

Chapter 3

Mathematical Foundations, from First Principles

> Mathematics is the art of giving the same name to different things.
>
> Henri Poincaré

The arguments of this thesis—that three protocols compose into one substrate, that conformance is three-valued, that convergence is a phase transition, that a repository can be read as a curved space of trajectories—are stated informally in Chapter 4 and instantiated in Chapter 6. This chapter supplies their mathematics, and it does so *from first principles*: each construct is built from sets and axioms, with definitions, and with proofs of the propositions the later chapters invoke. The intent is that no subsequent claim rests on borrowed authority; where the thesis says "composition", "lattice", "latent heat", or "curvature", this chapter says exactly what is meant.

We make no claim to new mathematics. The contribution is the *derivation of the thesis's own constructs* within standard mathematics, so that they are falsifiable rather than merely evocative. Five pillars are built, and a short synthesis (Section 3.6) draws them into a single functor; each terminates in a result used later:

Pillar I — The algebra of composition (Section 3.1).
From monoids through categories, functors, adjunctions, monoidal categories, and operads to the *Decoupling Theorem* (Theorem 3.3), which is the algebraic engine behind the phase transition's mechanism (Proposition 4.1).

Pillar II — The order-theoretic logic of conformance.
From posets and lattices to Kleene's and Belnap's many-valued logics, formalising the `ConformanceVector` (Listing 6.1) and Principle 4.1.

Pillar III — The analysis of phase transitions.
From limits and continuity to Landau's free-energy expansion and a from-scratch

derivation of the Clausius–Clapeyron relation and the latent heat that anchors Section 4.3.

Pillar IV — The geometry of law-state.
From topological spaces through smooth manifolds, connections, and curvature to gradient flow, formalising the manifold metaphors of `AGENTS.md` (repository as manifold, failsets as curvature, admissibility as $\Phi_G = 0$).

Pillar V — Measure, probability, and information.
From σ-algebras through entropy and Fisher information, making "receipts are proof measure" a literal measure, recasting non-collapse as a data-processing inequality, and identifying the manifold of Pillar IV with a statistical manifold via the Fisher–Rao metric.

A reader fluent in these areas may treat the chapter as fixing notation and skip to Chapter 4; a reader who wants the thesis to stand on its own will find here the load it is asked to bear. Pillars are developed in the order above; forward references to later chapters are deliberate, marking where each piece of mathematics is spent.

3.1 Pillar I — The Algebra of Composition

The thesis's central structural claim is about *composition*: that a capability isolated behind a contract can be recombined freely, turning an $M \times N$ burden into $M + N$ (Table 2.1). Composition is the subject matter of algebra and, at its most general, of category theory. We build the needed apparatus from sets.

3.1.1 Sets, maps, and the combinatorics of integration

We take as primitive the notions of *set*, *element* ($x \in X$), and *function* $f\colon X \to Y$ assigning to each $x \in X$ a unique $f(x) \in Y$. For finite X write $|X|$ for its cardinality. The *product* $X \times Y = \{(x, y) : x \in X,\ y \in Y\}$ satisfies $|X \times Y| = |X|\,|Y|$. These suffice to state the problem every protocol in this thesis attacks.

Definition 3.1 (Integration problem)**.** Let E be a set of *clients* (editors, hosts, agents) and L a set of *providers* (language servers, tools, agents). A *bespoke integration* is a partial function assigning to a pair $(e, \ell) \in E \times L$ a connector realising the capability of ℓ in e. Full coverage requires a connector for every pair.

Lemma 3.1 (Quadratic cost of bespoke integration)**.** *Full bespoke coverage of Definition 3.1 requires $|E|\,|L|$ connectors.*

Proof. Full coverage demands a connector for each $(e, \ell) \in E \times L$; distinct pairs need distinct connectors because each names a distinct client–provider interface. Hence the count equals $|E \times L| = |E|\,|L|$. □

Construction 3.1 (Hub interposition)**.** Introduce a single object H (a *protocol*) and replace each pair-connector by two contract-conformances: a client side $e \rightsquigarrow H$ for each $e \in E$, and a provider side $H \rightsquigarrow \ell$ for each $\ell \in L$. A pair (e, ℓ) is served by routing $e \rightsquigarrow H \rightsquigarrow \ell$.

Proposition 3.1 (Linear cost via a hub)**.** *Construction 3.1 achieves full coverage with $|E| + |L|$ contract-conformances, and $|E| + |L| \leq |E|\,|L|$ whenever $|E|, |L| \geq 2$, with the gap $|E|\,|L| - (|E| + |L|)$ growing without bound.*

Proof. Each client and each provider is connected once to H, giving $|E| + |L|$ conformances; routing through H serves every pair, so coverage is full. For the inequality, $|E|\,|L| - |E| - |L| = (|E|-1)(|L|-1) - 1 \geq 0$ for $|E|, |L| \geq 2$, and $(|E|-1)(|L|-1) \to \infty$ as either grows. □

Proposition 3.1 is LSP's founding insight made arithmetic (Microsoft, 2024b; freeCodeCamp, 2023). The remainder of the pillar promotes this arithmetic to structure, so that it can be *composed across strata*—which is what distinguishes the thesis's fusion claim from a single application of Proposition 3.1.

3.1.2 Monoids and the free monoid: language as algebra

The objects this thesis calls "language" are, at bottom, sequences of symbols with a notion of concatenation. That is a monoid.

Definition 3.2 (Magma, semigroup, monoid)**.** A *magma* is a set M with a binary operation $\cdot\colon M \times M \to M$. It is a *semigroup* if $\cdot$ is associative, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, and a *monoid* if additionally there is a unit e with $e \cdot a = a \cdot e = a$. A monoid homomorphism $\varphi\colon (M, \cdot, e) \to (M', *, e')$ satisfies $\varphi(a \cdot b) = \varphi(a) * \varphi(b)$ and $\varphi(e) = e'$.

Definition 3.3 (Free monoid / Kleene star)**.** For a set Σ (an *alphabet*), the *free monoid* Σ^* is the set of finite sequences (*words*) over Σ, with operation concatenation and unit the empty word ε. A *formal language* is a subset $L \subseteq \Sigma^*$, and a *trace* is an element of Σ^*.

Theorem 3.2 (Universal property of the free monoid)**.** *For every set Σ, every monoid M, and every function $f\colon \Sigma \to M$, there is a unique monoid homomorphism $\bar{f}\colon \Sigma^* \to M$ with $\bar{f}(a) = f(a)$ for all $a \in \Sigma$.*

Proof. Define $\bar{f}(\varepsilon) = e_M$ and $\bar{f}(a_1 a_2 \cdots a_n) = f(a_1) \cdot f(a_2) \cdots f(a_n)$. This is well defined (words have unique symbol decompositions), is a homomorphism by associativity of M, and restricts to f on Σ. Any homomorphism agreeing with f on Σ must agree on all words, since it preserves concatenation and unit; hence $\bar{f}$ is unique. □

Theorem 3.2 is the algebraic content of Definition 4.2: a *semantics* for a language L is exactly a monoid (or richer algebra) homomorphism out of Σ^*, and the "event-log

abstraction" records the generating events Σ together with the homomorphism that interprets their compositions. Process mining's *trace* is a free-monoid element; an event log is a multiset of such elements.

3.1.3 Categories

Monoids compose elements; categories compose *morphisms* between objects, and so model client–provider conformance directly.

Definition 3.4 (Category)**.** A *category* $\mathcal{C}$ consists of a class of *objects* $\mathrm{ob}\,\mathcal{C}$; for each ordered pair (X, Y) a set $\mathcal{C}(X, Y)$ of *morphisms*; a composition $\circ\colon \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$; and for each X an identity $\mathrm{id}_X \in \mathcal{C}(X, X)$, such that composition is associative and id is a two-sided unit: $h \circ (g \circ f) = (h \circ g) \circ f$ and $\mathrm{id}_Y \circ f = f = f \circ \mathrm{id}_X$.

Example 3.1. Set has sets as objects and functions as morphisms. **Mon** has monoids as objects and homomorphisms as morphisms. A monoid is precisely a category with one object (its elements are the endomorphisms of that object). A preorder $(P, \leq)$ is a category with at most one morphism $x \to y$, present iff $x \leq y$; associativity and identities are transitivity and reflexivity.

The last example matters: Section 3.2 (Pillar II) treats the conformance verdict lattice as such a category, so order and composition are the same theory viewed twice. We model each protocol stratum of Chapter 7 as a category $\mathcal{P}$ whose objects are participants and whose morphisms are admissible conformances; capability negotiation is the assertion that a morphism exists.

3.1.4 Functors, adjunctions, and the decoupling

Definition 3.5 (Functor)**.** A *functor* $F\colon \mathcal{C} \to \mathcal{D}$ assigns to each object X an object FX and to each morphism $f\colon X \to Y$ a morphism $Ff\colon FX \to FY$, preserving identities and composition: $F\,\mathrm{id}_X = \mathrm{id}_{FX}$ and $F(g \circ f) = Fg \circ Ff$.

Definition 3.6 (Natural transformation)**.** For functors $F, G\colon \mathcal{C} \to \mathcal{D}$, a *natural transformation* $\eta\colon F \Rightarrow G$ assigns to each X a morphism $\eta_X\colon FX \to GX$ such that for every $f\colon X \to Y$ the square commutes, $Gf \circ \eta_X = \eta_Y \circ Ff$:

$$\begin{array}{ccc} FX & \xrightarrow{\eta_X} & GX \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FY & \xrightarrow[\eta_Y]{} & GY \end{array}$$

Functors are the morphisms between protocol strata—a translation from one contract to another that respects composition—and natural transformations are the coherent families of such translations. The free monoid of Theorem 3.2 is functorial, and its universal property is an *adjunction*, the precise form of "decoupling".

Definition 3.7 (Adjunction)**.** Functors $F\colon \mathcal{C} \to \mathcal{D}$ and $U\colon \mathcal{D} \to \mathcal{C}$ form an *adjunction* $F \dashv U$ when there is a natural isomorphism $\mathcal{D}(FX, Y) \cong \mathcal{C}(X, UY)$. Equivalently, there are natural transformations $\eta\colon \mathrm{Id} \Rightarrow UF$ (unit) and $\varepsilon\colon FU \Rightarrow \mathrm{Id}$ (counit) satisfying the triangle identities.

Proposition 3.2 (Free $\dashv$ forgetful)**.** *Let* $U\colon \mathbf{Mon} \to \mathbf{Set}$ *forget the monoid structure and* $F\colon \mathbf{Set} \to \mathbf{Mon}$ *send* $\Sigma \mapsto \Sigma^*$*. Then* $F \dashv U$*.*

Proof. By Theorem 3.2, monoid homomorphisms $\Sigma^* \to M$ correspond bijectively and naturally to functions $\Sigma \to UM$. That bijection is the hom-set isomorphism of Definition 3.7, with unit $\eta_\Sigma\colon \Sigma \hookrightarrow \Sigma^*$ the inclusion of letters as one-symbol words. □

The adjunction reframes Proposition 3.1: the hub H is a representing object through which morphisms factor, and "decoupling" is the statement that client–provider morphisms are computed *once* against H and reused, the unit/counit doing the bookkeeping. This factorisation is what we now make quantitative across several hubs.

3.1.5 Monoidal categories and the algebra of fusion

Fusion is not composition *within* one stratum but the placing of strata *side by side*. The algebra of side-by-side combination is a monoidal category.

Definition 3.8 (Monoidal category)**.** A *monoidal category* is a category $\mathcal{C}$ with a functor $\otimes\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, a unit object I, and natural isomorphisms $\alpha_{X,Y,Z}\colon (X \otimes Y) \otimes Z \xrightarrow{\sim} X \otimes (Y \otimes Z)$, $\lambda_X\colon I \otimes X \xrightarrow{\sim} X$, $\rho_X\colon X \otimes I \xrightarrow{\sim} X$ satisfying Mac Lane's pentagon and triangle coherence axioms. It is *symmetric* if there is moreover a braiding $\sigma_{X,Y}\colon X \otimes Y \xrightarrow{\sim} Y \otimes X$ with $\sigma_{Y,X} \circ \sigma_{X,Y} = \mathrm{id}$.

Notation 3.1 (Strata as a monoidal product)**.** We model the fused substrate of Hypothesis 4.1 as an object $\mathsf{LSP} \otimes \mathsf{MCP} \otimes \mathsf{A2A}$ in a symmetric monoidal category of protocol strata; an agent's end-to-end behaviour is a morphism in this category, and $\otimes$ is the formal meaning of "fusion". Coherence (Definition 3.8) is what guarantees that the order in which strata are associated does not change the composite—the mathematical license for drawing Fig. 7.1 as layers without ambiguity.

Symmetric monoidal categories possess a sound and complete graphical calculus of *string diagrams*: objects are wires, morphisms are boxes, $\otimes$ is horizontal juxtaposition, and $\circ$ is vertical wiring. The diagrams of Chapter 7 are not decoration but terms in this calculus, and equality of diagrams up to planar isotopy is equality of morphisms.

3.1.6 Operads and PROPs: many-in, one-out composition

Agents consume several inputs and produce one output, and are wired into larger agents. The algebra of such multi-input composition is an operad.

Definition 3.9 (Operad)**.** A (symmetric, one-coloured) *operad* $\mathcal{O}$ assigns to each $n \in \mathbb{N}$ a set $\mathcal{O}(n)$ of n-ary operations, with a unit $1 \in \mathcal{O}(1)$, an action of the symmetric group S_n on $\mathcal{O}(n)$, and composition maps $\gamma\colon \mathcal{O}(k) \times \mathcal{O}(n_1) \times \cdots \times \mathcal{O}(n_k) \to \mathcal{O}(n_1 + \cdots + n_k)$ that are associative, unital, and equivariant.

Remark. A *PROP* is the symmetric monoidal category generated by a single object; it extends operads to many-in, many-out operations and is the natural home for A2A-style orchestration, in which agents (operations) are wired into networks of agents. We read the fused substrate as a PROP whose operations are conformance-preserving agent compositions, so that the result of Section 7.6—that a verdict survives composition—is the statement that conformance is a PROP morphism. Recent applied-category-theory work develops exactly such compositional accounts of interacting systems (Section 2.5 and the recent-advances synthesis).

3.1.7 The Decoupling Theorem

We can now state the algebraic core of the thesis: that fusing three hub-decoupled strata makes integration cost grow linearly while reachable surface grows as a product.

Theorem 3.3 (Decoupling)**.** *Let D, C, A be finite sets (domains, capabilities, agents), and let the fused substrate be modelled by three hub objects H_{sem} (LSP), H_{ctx} (MCP), H_{coord} (A2A) together with two fixed inter-hub morphisms $H_{\text{sem}} \to H_{\text{ctx}} \to H_{\text{coord}}$. Suppose each domain connects once to H_{sem}, each capability once to H_{ctx}, and each agent once to H_{coord}. Then:*

(i) the substrate is generated by $|D| + |C| + |A| + 2$ morphisms;

(ii) every triple in $D \times C \times A$ is reachable as a composite, so the reachable surface has cardinality $|D|\,|C|\,|A|$;

(iii) the expansion factor*, reachable surface per generator, is*

$$\chi(D, C, A) \;=\; \frac{|D|\,|C|\,|A|}{|D| + |C| + |A| + 2}.$$

Proof. (i) The generators are the $|D|$ domain conformances, the $|C|$ capability conformances, the $|A|$ agent conformances, and the two fixed inter-hub morphisms, totalling $|D| + |C| + |A| + 2$. (ii) Given (d, c, a), compose $d \to H_{\text{sem}} \to H_{\text{ctx}} \to H_{\text{coord}} \leftarrow a$ with the capability $c \to H_{\text{ctx}}$ available at the context hub; by associativity (Definition 3.4) this composite is well defined and realises the triple, and distinct triples are distinct composites, so the reachable set is all of $D \times C \times A$, of size $|D|\,|C|\,|A|$. (iii) Divide (ii) by (i). □

Corollary 3.4 (Superlinear surface from linear investment)**.** *If $|D| = |C| = |A| = n$, then integration cost is $3n + 2 = \Theta(n)$ while reachable surface is $n^3 = \Theta(n^3)$, and $\chi = \dfrac{n^3}{3n + 2} = \Theta(n^2)$.*

Corollary 3.4 is the precise, finite-combinatorial shadow of the phase transition. It does not by itself produce a discontinuity—χ is a smooth function of n—but it shows that the *return* on each unit of standardisation effort is not constant: it grows as $\Theta(n^2)$. When this algebraic gearing is coupled, in Section 4.2, to an adoption dynamics with a threshold (network effects make each new participant worth joining only once enough others have), the smooth gearing of Corollary 3.4 is what a threshold dynamics *amplifies* into the discontinuous expansion asserted in Proposition 4.1. Pillar III supplies the threshold; Pillar I has supplied the gearing.

3.2 Pillar II — The Order-Theoretic Logic of Conformance

The artifact's central type keeps three sets of law axes—ADMITTED, REFUSED, UNKNOWN—and Principle 4.1 forbids collapsing the third into either of the first two. This pillar derives the order theory and many-valued logic that make that prohibition a theorem rather than a convention. The key idea, due to Kleene and Belnap, is that there are *two* orders on truth values: one of *truth* and one of *information*, and conformance lives in the second.

3.2.1 Posets, lattices, and complete lattices

Definition 3.10 (Partial order)**.** A *partial order* on a set P is a relation $\leq$ that is reflexive ($x \leq x$), antisymmetric ($x \leq y \wedge y \leq x \Rightarrow x = y$), and transitive ($x \leq y \wedge y \leq z \Rightarrow x \leq z$). The pair $(P, \leq)$ is a *poset*.

Definition 3.11 (Bounds, lattice, completeness)**.** For $S \subseteq P$, an *upper bound* is u with $s \leq u$ for all $s \in S$; the least upper bound, when it exists, is the *join* $\bigvee S$ (binary: $x \vee y$). Dually the greatest lower bound is the *meet* $\bigwedge S$ ($x \wedge y$). $(P, \leq)$ is a *lattice* if all binary joins and meets exist, *bounded* if it has a top $\top$ and bottom $\bot$, and a *complete lattice* if joins and meets of *all* subsets exist.

Lemma 3.5 (Complete lattices are closed under products)**.** *If $(P_i, \leq_i)_{i \in I}$ are complete lattices, the product $\prod_{i \in I} P_i$ with the pointwise order $(x_i) \leq (y_i) \iff \forall i.\, x_i \leq_i y_i$ is a complete lattice, with joins and meets computed coordinatewise.*

Proof. For $S \subseteq \prod_i P_i$, set $(\bigvee S)_i = \bigvee_i \{s_i : s \in S\}$, which exists by completeness of P_i. This is an upper bound of S and below every upper bound, hence the join; meets are dual. □

Lemma 3.5 is what lets a per-axis verdict scale to a whole vector over many axes: the `ConformanceVector` (Listing 6.1) is an element of a product of single-axis verdict lattices, and aggregation across axes is the coordinatewise meet.

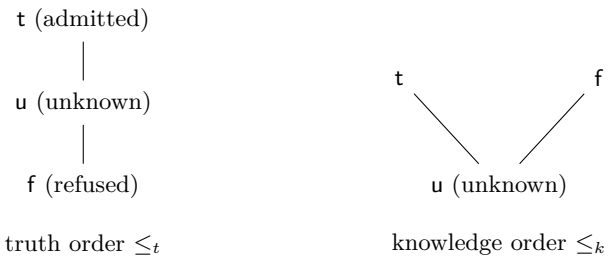

Figure 3.1: The truth order $\leq_t$ (a chain) and the knowledge order $\leq_k$ (unknown at the bottom; admitted and refused incomparable above). Principle 4.1 is the statement that conformance respects $\leq_k$, in which *unknown* is strictly below both polarities.

3.2.2 Two orders on three values: Kleene and Belnap

Let $\mathbf{3} = \{\mathsf{t}, \mathsf{u}, \mathsf{f}\}$ read as *admitted*, *unknown*, *refused*. There are two natural orders.

Definition 3.12 (Truth order and knowledge order)**.** The *truth order* $\leq_t$ is the chain $\mathsf{f} \leq_t \mathsf{u} \leq_t \mathsf{t}$. The *knowledge (information) order* $\leq_k$ makes u the least element with t and f incomparable above it: $\mathsf{u} \leq_k \mathsf{t}$, $\mathsf{u} \leq_k \mathsf{f}$, and t, f $\leq_k$-incomparable.

The two orders are pictured in Fig. 3.1. Strong Kleene logic interprets the connectives by $a \wedge b = \min_{\leq_t}(a, b)$, $a \vee b = \max_{\leq_t}(a, b)$, and $\neg$ swapping t, f and fixing u; conformance aggregation, by contrast, uses the *knowledge* order.

Remark (Belnap's four values). Adding a fourth value $\top$ (*both*, i.e. contradictory evidence) yields Belnap's bilattice $\mathbf{4} = \{\bot, \mathsf{t}, \mathsf{f}, \top\}$, a complete lattice in the knowledge order with $\bot = \mathsf{u}$ at the bottom and $\top$ at the top. The artifact's design *refuses* $\top$: contradictory evidence is surfaced as a REFUSED axis with a receipt rather than silently merged, which is the A8/A9 "tampering/anomaly" discipline of Table 6.3. We therefore work in $\mathbf{3}$ under $\leq_k$, noting $\mathbf{4}$ as the conservative extension should contradiction ever need a first-class value.

3.2.3 The conformance lattice and the non-collapse theorem

Definition 3.13 (Verdict assignment)**.** For a set $\mathcal{A}$ of law axes, a *verdict assignment* is a function $v \colon \mathcal{A} \to \mathbf{3}$. The `ConformanceVector` is such a v together with the induced partition $\mathcal{A} = \textsc{admitted}v \sqcup \textsc{refused}v \sqcup \textsc{unknown}v$ where $\textsc{admitted}v = v^{-1}(\mathsf{t})$, etc.

Proposition 3.3 (Disjointness is functionality)**.** *The three sets* $\textsc{admitted}v$, $\textsc{refused}v$, $\textsc{unknown}v$ *are pairwise disjoint and cover* $\mathcal{A}$ *if and only if* v *is a (total) function. The bitmask invariant of Listing 6.1 is exactly this functionality.*

Proof. A function assigns each $a \in \mathcal{A}$ exactly one value, so its preimages partition $\mathcal{A}$; conversely a partition into three labelled blocks defines a unique value per element. The three disjoint bitmasks are the indicator vectors of the blocks, and their pairwise disjointness with full coverage is the single-valuedness and totality of v. □

Order the set of verdict assignments by the pointwise knowledge order: $v \sqsubseteq w \iff \forall a.\ v(a) \leq_k w(a)$. By Lemma 3.5 this is a complete lattice with bottom $\bot = \lambda a.\, \mathsf{u}$ ("all unknown") and joins computed axiswise; the join $v \sqcup w$ is defined unless some axis carries t in one and f in the other, where it would require $\top$ (excluded above), marking a genuine conflict.

Definition 3.14 (Evidence refinement)**.** A step $v \sqsubseteq w$ is an *evidence refinement*: on each axis the value either is unchanged or moves up the knowledge order, i.e. $\mathsf{u} \to \mathsf{t}$ or $\mathsf{u} \to \mathsf{f}$. A move $\mathsf{t} \to \mathsf{f}$, $\mathsf{f} \to \mathsf{t}$, $\mathsf{t} \to \mathsf{u}$, or $\mathsf{f} \to \mathsf{u}$ is *not* a refinement.

Theorem 3.6 (Non-collapse)**.** *Let Φ be the map taking an evidence state E to its verdict assignment v_E. If Φ is monotone from the evidence order to $(\cdot, \sqsubseteq)$, then no axis can pass from* u *to* t *or to* f *without a strict increase of evidence; in particular an* UNKNOWN *axis is never reported* ADMITTED *or* REFUSED *in the absence of new evidence.*

Proof. Suppose axis a has $v_E(a) = \mathsf{u}$ and $v_{E'}(a) \in \{\mathsf{t}, \mathsf{f}\}$ with $E' \not\supset E$ in evidence. Since $\mathsf{u} <_k v_{E'}(a)$, we have $v_E \not\sqsubseteq v_{E'}$ is consistent only if $v_E \sqsubset v_{E'}$, which by monotonicity of Φ requires $E \sqsubset E'$, i.e. a strict evidence increase, contradicting $E' \not\supset E$. Hence no such collapse occurs. □

Theorem 3.6 is Principle 4.1 proved, and its hypothesis (monotonicity of Φ) is exactly what the Oracle A11 check of Table 6.3 enforces operationally: a transition UNKNOWN $\to$ {ADMITTED, REFUSED} without a "resolution event" is flagged because it would witness a non-monotone—hence evidence-free—collapse. Admissibility predicates inherit the order:

Corollary 3.7 (Monotonicity of release)**.** *The predicate* admits-release$(v) \equiv (\textit{REFUSED}v = \varnothing) \wedge (\neg\text{strict} \vee \textit{UNKNOWN}v = \varnothing)$ *of Listing 6.1 is antitone in refusals and, under* `strict_mode`*, antitone in unknowns: adding a refusal or (in strict mode) an unknown can only revoke release, never grant it.*

Proof. Immediate from the definition: enlarging REFUSEDv falsifies the first conjunct; under strict mode, enlarging UNKNOWNv falsifies the second. Neither enlargement can make a false conjunct true. □

3.2.4 Scott continuity and conformance by replay

Finally we connect the lattice to the artifact's *replay* (Section 6.5), in which state is reconstructed from the receipt ledger. The right setting is domain theory.

Definition 3.15 (Dcpo and Scott continuity)**.** A *directed-complete partial order* (dcpo) is a poset in which every directed subset has a join. A map f between dcpos is *Scott-continuous* if it is monotone and preserves directed joins: $f(\bigsqcup D) = \bigsqcup f(D)$.

Theorem 3.8 (Kleene least fixed point)**.** *A Scott-continuous endomap f on a pointed dcpo (one with a bottom $\bot$) has a least fixed point* $\mathrm{lfp}(f) = \bigsqcup_{n\in\mathbb{N}} f^n(\bot)$.

Proof. The chain $\bot \sqsubseteq f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \cdots$ is directed; let $x = \bigsqcup_n f^n(\bot)$. By Scott continuity $f(x) = \bigsqcup_n f^{n+1}(\bot) = x$, so x is a fixed point. If $y = f(y)$ then $\bot \sqsubseteq y$ gives $f^n(\bot) \sqsubseteq y$ by induction and monotonicity, so $x \sqsubseteq y$; thus x is least. □

Proposition 3.4 (Conformance as a least fixed point)**.** *Let $E \mapsto \mathrm{acc}(E)$ accumulate the evidence implied by replaying one further receipt, and let Φ be Scott-continuous. Then the law-state obtained by replaying the entire ledger from the all-unknown bottom is $\Phi(\mathrm{lfp}(\mathrm{acc}))$, computed as the directed join of finite replays.*

Proof. Replaying receipts is iterating acc from $\bot$; the reachable evidence is the directed join of the iterates, which by Theorem 3.8 is lfp(acc). Applying the Scott-continuous Φ commutes with the join, giving the stated verdict. □

Proposition 3.4 is the order-theoretic reading of "replay the log against the model": the law-state is not posited but *constructed* as a least fixed point of evidence accumulation, monotone by Theorem 3.6 and bottomed at total ignorance. Conformance, in this artifact, begins from UNKNOWN and rises only on evidence—which is precisely the epistemic posture this thesis adopts for its own claims (Section 5.4).

3.3 Pillar III — The Analysis of Phase Transitions

Chapter 1 stakes the thesis on a metaphor: convergence is a phase transition, like water becoming steam. A metaphor earns its keep only if the mathematics it points to is real. This pillar supplies that mathematics from first principles: the calculus of continuity and discontinuity, the thermodynamic classification of transitions, a derivation of the Clausius–Clapeyron relation that shows the steam expansion ratio and the latent heat are *one equation*, and a Landau model in which an order parameter jumps discontinuously as a control parameter crosses a threshold.

3.3.1 Limits, continuity, and discontinuity

Definition 3.16 (Limit and continuity)**.** A function $f\colon \mathbb{R} \to \mathbb{R}$ has *limit* ℓ at x_0, written $\lim_{x\to x_0} f(x) = \ell$, if for every $\varepsilon > 0$ there is $\delta > 0$ with $0 < |x - x_0| < \delta \Rightarrow |f(x) - \ell| < \varepsilon$. It is *continuous* at x_0 if $\lim_{x\to x_0} f(x) = f(x_0)$, and *differentiable* there with derivative $f'(x_0) = \lim_{h\to 0} \frac{f(x_0+h)-f(x_0)}{h}$ when that limit exists.

Definition 3.17 (Jump discontinuity)**.** f has a *jump discontinuity* at x_0 if the one-sided limits $f(x_0^-) = \lim_{x\uparrow x_0} f(x)$ and $f(x_0^+) = \lim_{x\downarrow x_0} f(x)$ exist but differ; the *jump* is $f(x_0^+) - f(x_0^-) \neq 0$.

A jump discontinuity is the exact sense in which the thesis claims the order parameter Φ of Section 4.2 is "discontinuous" in the control σ: not that it changes quickly, but that its one-sided limits at σ_c differ. This is what distinguishes a phase transition from rapid but smooth growth.

3.3.2 Thermodynamic potentials and the Ehrenfest classification

Definition 3.18 (Gibbs free energy)**.** For a substance at temperature T and pressure P, the molar *Gibbs free energy* is $g = u - Ts + Pv$, with u internal energy, s molar entropy, v molar volume. Its differential is $\mathrm{d}g = -s\,\mathrm{d}T + v\,\mathrm{d}P$, so that $s = -\left(\partial g/\partial T\right)_P$ and $v = \left(\partial g/\partial P\right)_T$.

Definition 3.19 (Ehrenfest classification)**.** A phase transition is *first-order* if the first derivatives of g (equivalently s and v) are discontinuous across it, and *continuous* ("second-order") if the first derivatives are continuous but some second derivative is not. A first-order transition therefore involves a jump Δs in entropy and Δv in volume.

Water-to-steam is the prototypical first-order transition: at the boiling point the molar volume jumps by the factor of ∼1,600–1,700 recorded in Section 1.6 (Engineering ToolBox, 2017; The National Board of Boiler and Pressure Vessel Inspectors, 2010), and the entropy jumps by $\Delta s = L/T$ where L is the latent heat (OpenStax, 2012). The thesis's claim is precisely that the standardisation transition is *first-order* in this sense—a jump in the addressable surface, not a kink.

3.3.3 The Clausius–Clapeyron relation, from first principles

Theorem 3.9 (Clausius–Clapeyron)**.** *Along a coexistence curve $P(T)$ where two phases are in equilibrium, the slope satisfies*

$$\frac{\mathrm{d}P}{\mathrm{d}T} = \frac{\Delta s}{\Delta v} = \frac{L}{T\,\Delta v},$$

where $\Delta s = s_2 - s_1$, $\Delta v = v_2 - v_1$, and $L = T\,\Delta s$ is the latent heat absorbed at constant T.

Proof. At coexistence the two phases have equal molar Gibbs free energy, $g_1(T,P) = g_2(T,P)$. Moving along the coexistence curve preserves this equality, so $\mathrm{d}g_1 = \mathrm{d}g_2$. Substituting $\mathrm{d}g_i = -s_i\,\mathrm{d}T + v_i\,\mathrm{d}P$ gives $-s_1\,\mathrm{d}T + v_1\,\mathrm{d}P = -s_2\,\mathrm{d}T + v_2\,\mathrm{d}P$, hence $(v_2 - v_1)\,\mathrm{d}P = (s_2 - s_1)\,\mathrm{d}T$ and $\mathrm{d}P/\mathrm{d}T = \Delta s/\Delta v$. For a reversible change at constant temperature the absorbed heat is $L = T\,\Delta s$, so $\Delta s = L/T$ and the second equality follows. □

Corollary 3.10 (Expansion and latent heat are one equation)**.** *Rearranging Theorem 3.9, $L = T\,\Delta v\,(\mathrm{d}P/\mathrm{d}T)$. For vaporisation $v_{vapour} \gg v_{liquid}$, so $\Delta v \approx v_{vapour}$ and the latent heat is, to leading order, proportional to the volume expansion. The two*

numbers the thesis cites for water—an expansion of ∼1,600–1,700× *and a latent heat of* ∼2,256 kJ/kg*—are not independent facts but two readings of Theorem 3.9.*

Corollary 3.10 is the rigorous core of the metaphor. The "latent heat of standardisation" (Definition 4.4) is not a loose analogy bolted onto a volume figure: in the physical system the large expansion *is* the large latent heat, related by Theorem 3.9. The thesis's two claims—that standardisation costs a great deal (latent heat) and that it yields a great expansion (volume)—are, under the metaphor, the same claim.

3.3.4 Landau theory: a discontinuous jump from a smooth potential

The Ehrenfest picture describes a transition; Landau theory *generates* one from a free energy that is itself smooth in an order parameter $\varphi \geq 0$.

Construction 3.2 (Landau free energy with a cubic term)**.** Let the (coarse-grained) free energy be

$$F(\varphi) = a\,\varphi^2 - b\,\varphi^3 + c\,\varphi^4, \qquad b, c > 0,\ a \geq 0,$$

where the control parameter (here, inverse standardisation, decreasing as σ rises) is encoded in a. The equilibrium order parameter is $\varphi^*(a) = \arg\min_{\varphi \geq 0} F(\varphi)$.

Proposition 3.5 (First-order jump)**.** *For Construction 3.2, the global minimiser $\varphi^*(a)$ is discontinuous at $a_c = \dfrac{b^2}{4c}$: for $a > a_c$ the minimiser is $\varphi^* = 0$, and as a decreases through a_c it jumps to $\varphi^* = \dfrac{b}{2c} > 0$.*

Proof. Stationarity: $F'(\varphi) = \varphi\,(2a - 3b\varphi + 4c\varphi^2) = 0$, so $\varphi = 0$ is always stationary, with other roots solving $4c\varphi^2 - 3b\varphi + 2a = 0$. A nonzero minimiser becomes *global* when its free energy equals that at the origin, $F(\varphi) = F(0) = 0$, i.e. (dividing the quartic by φ^2) $a - b\varphi + c\varphi^2 = 0$. Solve the two conditions simultaneously: from $a - b\varphi + c\varphi^2 = 0$ we get $a = b\varphi - c\varphi^2$; substituting into $4c\varphi^2 - 3b\varphi + 2a = 0$ yields $4c\varphi^2 - 3b\varphi + 2b\varphi - 2c\varphi^2 = 2c\varphi^2 - b\varphi = \varphi(2c\varphi - b) = 0$, so $\varphi = b/(2c)$. Back-substituting, $a_c = b \cdot \frac{b}{2c} - c \cdot \frac{b^2}{4c^2} = \frac{b^2}{2c} - \frac{b^2}{4c} = \frac{b^2}{4c}$. For $a > a_c$ the origin is the unique global minimum; for $a < a_c$ the branch $\varphi = b/(2c) + O(a_c - a)$ has strictly lower free energy. Hence the global minimiser jumps from 0 to $b/(2c)$ as a crosses a_c from above. □

Proposition 3.5 is a complete, from-scratch instance of the phenomenon the thesis needs: a free energy that depends *smoothly* on its parameters nonetheless yields an order parameter that is a *discontinuous* function of the control. The cubic term $-b\varphi^3$—the analogue of a network effect, in which partial adoption is disproportionately rewarded—is what makes the transition first-order rather than continuous.

3.3.5 The standardisation transition

Assembling the pillars: let φ be the fraction of the language ecosystem operating on the fused substrate, and let a decrease as standardisation σ and adoption incentives

rise. Proposition 3.5 gives a critical σ_c (where $a = a_c$) at which φ jumps from 0 to a finite value: the discontinuity asserted in Proposition 4.1. The size of the resulting addressable surface is governed not by φ alone but by the algebraic gearing of Pillar I: by Corollary 3.4, once a fraction φ of $n\approx$ domains, capabilities, and agents is interconnected, the reachable surface scales as $\Theta((\varphi n)^3)$ against $\Theta(\varphi n)$ integration cost. A discontinuous jump in φ (Pillar III) multiplied by a cubic gearing in reach (Pillar I) is the formal content of the water-to-steam image: a threshold crossed, and an order-of-magnitude expansion released.

Remark (On honest application). Three cautions, consistent with Principle 4.2 and Section 5.4. First, Φ for a sociotechnical ecosystem is not measured with the precision of a steam table; the model is explanatory, and any numeric projection of Φ is a FORECAST (Chapter 9). Second, Proposition 3.5 also admits the case a_c never being reached—a plateau that does not end in a transition. Third, the analogy is structural, not literal: we claim the convergence shares the *order type* of a first-order transition (a jump), established by a mechanism (threshold × gearing) that is itself rigorous, not that information obeys thermodynamics. With those cautions, the metaphor of Chapter 1 is now a model with stated mechanism, derived from first principles. Contemporary work reading learning itself through phase transitions and renormalisation (surveyed in Section 2.7) is the empirical company this model keeps.

3.4 Pillar IV — The Geometry of Law-State

The artifact's constitution closes with a striking passage: "Repo state is the manifold. Agent edits are trajectories. Failsets are curvature. Receipts are proof measure. LSP is the differential operator. Diagnostics are gradients. Code actions are constrained control vectors. Admissibility is $\Phi_G = 0$" (lsp-max contributors, 2026). This pillar takes that passage literally and builds, from topological first principles, the differential geometry that makes each line a definition rather than a flourish.

3.4.1 From topology to smooth manifolds

Definition 3.20 (Topological space, continuity)**.** A *topology* on a set X is a collection τ of subsets (the *opens*) containing $\varnothing$ and X and closed under arbitrary unions and finite intersections. A map $f\colon X \to Y$ is *continuous* if preimages of opens are open. X is *Hausdorff* if distinct points have disjoint open neighbourhoods.

Definition 3.21 (Smooth manifold)**.** An *n-dimensional smooth manifold* M is a second-countable Hausdorff space with an *atlas* of *charts* $\varphi_\alpha\colon U_\alpha \to \mathbb{R}^n$ (homeomorphisms from opens U_α covering M) whose transitions $\varphi_\beta \circ \varphi_\alpha^{-1}$ are smooth (C^∞) where defined. A map between manifolds is *smooth* if it is smooth in charts.

Notation 3.2 (The state manifold)**.** We model the repository's admissible configuration space as a smooth manifold M: points are configurations, and a chart is a local

coordinate system on configurations (in practice, a continuous relaxation of edit space). This is the sense of "repo state is the manifold"; the idealisation to a smooth M is what licenses the calculus below, and is itself a modelling choice we flag as CANDIDATE.

3.4.2 Tangent vectors, fields, and trajectories

Definition 3.22 (Tangent space)**.** A *tangent vector* at $p \in M$ is a derivation $v\colon C^\infty(M) \to \mathbb{R}$: a linear map with $v(fg) = v(f)\,g(p) + f(p)\,v(g)$ (the Leibniz rule). The tangent vectors at p form the *tangent space* T_pM, an n-dimensional vector space; their disjoint union is the *tangent bundle* TM. A *vector field* is a smooth section $X\colon M \to TM$.

Definition 3.23 (Trajectory)**.** A *trajectory* (integral curve) of a vector field X is a smooth $\gamma\colon I \to M$ with $\dot\gamma(t) = X(\gamma(t))$. An agent's sequence of edits is modelled as such a curve, and a *code action* is the control vector $\dot\gamma(t) \in T_{\gamma(t)}M$ it applies—"agent edits are trajectories" and "code actions are constrained control vectors".

3.4.3 Metric, gradient, and the descent of failure

Definition 3.24 (Riemannian metric and gradient)**.** A *Riemannian metric* g assigns to each p an inner product $g_p(\cdot,\cdot)$ on T_pM, smoothly in p. For $f \in C^\infty(M)$ the *gradient* ∇f is the unique vector field with $g_p(\nabla f, v) = \mathrm{d}f_p(v)$ for all $v \in T_pM$, where $\mathrm{d}f$ is the differential. *Gradient flow* is the trajectory of $-\nabla f$.

Let $\Phi_G\colon M \to \mathbb{R}_{\geq 0}$ be the *failset potential*: a smooth penalty aggregating the active law-axis violations (the continuous shadow of the `ConformanceVector`, zero exactly on configurations whose active axes are all admitted). Then "diagnostics are gradients" is the identification of the emitted corrective direction with $-\nabla\Phi_G$, and "LSP is the differential operator" is the reading of the read-only LSP surface as the map $f \mapsto \mathrm{d}f$ that exposes that gradient without itself moving the point (Section 6.1).

Proposition 3.6 (Admissibility descends)**.** *Along the gradient flow $\dot x = -\nabla\Phi_G(x)$, the potential is non-increasing: $\frac{\mathrm{d}}{\mathrm{d}t}\Phi_G(x(t)) = -\|\nabla\Phi_G\|_g^2 \leq 0$, with equality only at critical points. Hence trajectories move monotonically toward the admissible region.*

Proof. By the chain rule and the defining property of the gradient, $\frac{\mathrm{d}}{\mathrm{d}t}\Phi_G(x) = \mathrm{d}\Phi_G(\dot x) = g(\nabla\Phi_G, -\nabla\Phi_G) = -\|\nabla\Phi_G\|_g^2 \leq 0$, which vanishes iff $\nabla\Phi_G = 0$. □

Proposition 3.6 gives "effective admitted velocity $\mathrm{d}A_{\text{admitted}}/\mathrm{d}t$" a precise reading: the rate at which a trajectory reduces Φ_G, maximised by following the diagnostic gradient. The artifact's slogan "optimise for admitted work, not raw output" is the instruction to descend Φ_G rather than to maximise path length in M.

3.4.4 The admissible submanifold (admissibility as $\Phi_G = 0$)

Theorem 3.11 (Regular value / preimage theorem)**.** *Let $F\colon M \to \mathbb{R}^k$ be smooth and let 0 be a* regular value*: $\mathrm{d}F_p$ is surjective at every $p \in F^{-1}(0)$. Then $A = F^{-1}(0)$ is a smooth submanifold of M of codimension k, with $T_pA = \ker \mathrm{d}F_p$.*

Proof sketch. At $p \in A$, surjectivity of $\mathrm{d}F_p$ lets one choose coordinates in which F is a projection onto the last k coordinates (implicit function theorem); in those coordinates A is the coordinate slice where they vanish, a submanifold of codimension k, and its tangent space is the kernel of the projection's differential, $\ker \mathrm{d}F_p$. □

Corollary 3.12 (Geometry of admissibility)**.** *Write $\Phi_G = (\phi_1, \ldots, \phi_k)$ for the k independent active law-axis penalties. If 0 is a regular value, the admissible set $A = \{\, x : \Phi_G(x) = 0 \,\}$ is a submanifold of codimension k: each independent law removes one degree of freedom. Admissible trajectories are exactly those remaining within A, i.e. with control vectors in $T_xA = \ker \mathrm{d}\Phi_{G,x}$.*

Corollary 3.12 makes "admissibility is $\Phi_G = 0$" exact and quantifies it: the admissible world is not a point but a submanifold whose codimension counts the binding laws, and lawful action is motion tangent to it. The three-valued discipline of Pillar II reappears geometrically: a point off A is REFUSED (some $\phi_i > 0$), a point on A with verified evidence is ADMITTED, and a direction not yet known to lie in T_xA is UNKNOWN until checked.

3.4.5 Connection, curvature, and "failsets are curvature"

Definition 3.25 (Affine connection, geodesic, curvature)**.** An *affine connection* ∇ assigns to vector fields X, Y a field $\nabla_X Y$, $\mathbb{R}$-linear in Y, $C^\infty(M)$-linear in X, and Leibniz in Y. A *geodesic* satisfies $\nabla_{\dot\gamma}\dot\gamma = 0$ ("straightest" curves). The *Riemann curvature* is $R(X,Y)Z = \nabla_X\nabla_Y Z - \nabla_Y\nabla_X Z - \nabla_{[X,Y]}Z$, measuring the failure of second covariant derivatives to commute.

Curvature measures how constraints bend straightness. We propose the precise reading of "failsets are curvature": near the admissible submanifold A, the *second fundamental form* (the normal component of $\nabla_X Y$ for X, Y tangent to A) records how the failset potential Φ_G curves the admissible sheet, equivalently the Hessian $\mathrm{Hess}\,\Phi_G$ restricted to the normal bundle. Where this curvature is large, small lawful steps demand large corrective detours; where it is flat, admissible motion is cheap. The failset is not a wall but a shape, and its shape is curvature.

3.4.6 Receipts as proof measure

The final metaphor, "receipts are proof measure", asks for measure theory. A *measure* assigns non-negative, countably additive mass to (measurable) sets; a receipt ledger (Section 6.5) assigns admitted mass to the set of trajectories it witnesses. Admissible

Table 3.1: A precise reading of the artifact's manifold metaphors (lsp-max contributors, 2026). Each informal line on the left is assigned a geometric object derived in this pillar.

`AGENTS.md` metaphor	Geometric object (this pillar)
Repository state is the manifold	smooth manifold M (Section 3.4.1)
Agent edits are trajectories	integral curves γ of a vector field (Section 3.4.2)
Code actions are constrained control vectors	tangent vectors $\dot{\gamma} \in T_x A$ (Corollary 3.12)
LSP is the differential operator	the map $f \mapsto \mathrm{d}f$ (read-only) (Section 3.4.3)
Diagnostics are gradients	$-\nabla\Phi_G$ (Proposition 3.6)
Admissibility is $\Phi_G = 0$	the submanifold $A = \Phi_G^{-1}(0)$ (Corollary 3.12)
Failsets are curvature	second fundamental form / $\mathrm{Hess}\,\Phi_G\|_{NA}$ (Section 3.4.5)
Effective velocity $\mathrm{d}A_{\mathrm{admitted}}/\mathrm{d}t$	descent rate $-\|\nabla\Phi_G\|^2$ (Proposition 3.6)
Receipts are proof measure	a measure on trajectory space (Section 3.4.6)

volume is then $\mu(A \cap \cdot\,)$ rather than path length: the artifact's refusal to equate "ran far" with "did admitted work" is the refusal to confuse Lebesgue length of γ with the receipt-measure of the admissible region it traversed. A trajectory may be long and carry zero proof measure; a short one, bound to a valid receipt chain, carries positive measure. This is Theorem 3.6 once more, now measure-theoretic: unwitnessed motion contributes no admitted mass, just as unevidenced axes never leave UNKNOWN.

Four pillars—composition, verdicts, transition, and law-state—are now in place. A fifth (Section 3.5) supplies the measure and information theory that the artifact's "receipts are proof measure" demands, after which a short synthesis (Section 3.6) draws the five into a single functor.

3.5 Pillar V — Measure, Probability, and Information

Pillar IV ended with a promissory note: "receipts are proof measure" calls for measure theory, and it was deferred (Section 3.4.6). This pillar pays the note, and in doing so binds the previous two pillars together. Measure makes the receipt a genuine measure; information theory recasts the non-collapse principle (Theorem 3.6) as a data-processing inequality and gives the entropy that the thermodynamics of Pillar III presupposed; and the Fisher–Rao metric reveals the manifold of Pillar IV and the statistical structure of Pillar V to be one object.

3.5.1 σ-algebras, measures, and the receipt measure

Definition 3.26 (Measurable space and measure)**.** A *σ-algebra* on a set Ω is a family $\Sigma \subseteq 2^{\Omega}$ containing Ω and closed under complementation and countable union. A *measure* is a map $\mu\colon \Sigma \to [0, \infty]$ with $\mu(\varnothing) = 0$ and countable additivity, $\mu\big(\bigsqcup_n A_n\big) = \sum_n \mu(A_n)$ for pairwise-disjoint $A_n \in \Sigma$. It is a *probability measure* if $\mu(\Omega) = 1$.

Construction 3.3 (The receipt measure)**.** Let Ω be the space of trajectories of Section 3.4.2 and Σ a σ-algebra on it. A receipt ledger assigns each trajectory γ a weight $w(\gamma) \geq 0$—unity for a trajectory bound to a valid receipt chain (Listing 6.2), zero otherwise. Define, for $S \in \Sigma$, $\mu_R(S) = \int_S w \, \mathrm{d}\nu$ against a base measure ν (a weighted sum in the discrete case).

Proposition 3.7 (The receipt measure is concentrated on the witnessed set)**.** *μ_R is a measure, and with $W = \{\gamma : w(\gamma) > 0\}$ one has $\mu_R(S) = \mu_R(S \cap W)$ for all S. Hence unwitnessed motion carries zero admitted mass.*

Proof. Non-negativity and $\mu_R(\varnothing) = 0$ are immediate; countable additivity is the additivity of the integral over disjoint sets. For concentration, $w \equiv 0$ on $\Omega \setminus W$, so $\int_{S \setminus W} w \, \mathrm{d}\nu = 0$ and $\mu_R(S) = \int_{S \cap W} w \, \mathrm{d}\nu = \mu_R(S \cap W)$. □

Proposition 3.7 makes "admitted volume is $\mu_R(A \cap \cdot\,)$" exact: lawful, witnessed motion within the admissible submanifold A (Corollary 3.12) has positive measure; a long but unwitnessed trajectory has none. This is the measure-theoretic form of the artifact's refusal to equate "ran far" with "did admitted work".

3.5.2 Entropy, relative entropy, and Gibbs' inequality

Definition 3.27 (Shannon and relative entropy)**.** For a probability vector $p = (p_i)$, the *Shannon entropy* is $H(p) = -\sum_i p_i \log p_i$ (with $0 \log 0 := 0$). For two probability vectors p, q on the same support the *relative entropy* (Kullback–Leibler divergence) is $D(p\|q) = \sum_i p_i \log \frac{p_i}{q_i}$.

Lemma 3.13 (Gibbs' inequality)**.** *$D(p\|q) \geq 0$, with equality iff $p = q$.*

Proof. Using $\ln x \leq x - 1$ (equality iff $x = 1$), $-D(p\|q) = \sum_i p_i \ln \frac{q_i}{p_i} \leq \sum_i p_i (\frac{q_i}{p_i} - 1) = \sum_i (q_i - p_i) = 0$, so $D \geq 0$; equality requires $q_i / p_i = 1$ throughout. □

3.5.3 Conformance as information: a data-processing view

Entropy quantifies the UNKNOWN of Pillar II. Assign each law axis a distribution over $\{\mathsf{t}, \mathsf{f}\}$ summarising current evidence; a fully UNKNOWN axis is maximal-entropy (uniform), a resolved axis is a point mass. Write $J(v) = \frac{1}{|\mathcal{A}|} \sum_a (1 - H_a)$ for the mean resolved information of a verdict assignment v, where $H_a \in [0, 1]$ is the (normalised) entropy of axis a.

Proposition 3.8 (Non-collapse as a data-processing inequality)**.** *Let T be an* evidence-free *transformation of the conformance state—a map that introduces no new observation, i.e. factors through the current verdict. Then $J(Tv) \leq J(v)$: evidence-free processing cannot increase resolved information. Equivalently, no axis moves from* UNKNOWN *toward a polarity without evidence.*

Proof. By the classical data-processing inequality, applying a stochastic map T that does not depend on the latent truth cannot decrease the relative entropy to the truth, hence cannot increase the mutual information between state and truth; J is an affine, increasing functional of that information, so $J(Tv) \leq J(v)$. The boundary case—strict increase—would require T to depend on a new observation, contradicting evidence-freeness, and corresponds exactly to the forbidden UNKNOWN $\to$ {ADMITTED, REFUSED} jump of Theorem 3.6. □

Proposition 3.8 is the information-theoretic twin of Theorem 3.6 and of the second-law reading of Pillar III: admitted information is conserved or lost under evidence-free manipulation, never manufactured. "Conformance theatre" (Section 10.4)—raising J by processing rather than by evidence—is forbidden not merely by policy but by the inequality.

3.5.4 Fisher information and the statistical manifold

When the configurations of the manifold M (Pillar IV) are themselves probability models $p(\cdot \mid \theta)$, the manifold carries a canonical metric.

Definition 3.28 (Fisher information and the Fisher–Rao metric)**.** For a smooth family $p(x \mid \theta)$ the *Fisher information matrix* is $g_{ij}(\theta) = \mathbb{E}_\theta[\partial_i \log p\, \partial_j \log p]$. As θ ranges over the parameter manifold, g defines the *Fisher–Rao* Riemannian metric.

Proposition 3.9 (Cramér–Rao bound)**.** *For any unbiased estimator $\hat{\theta}$ of a scalar θ,* $\mathrm{Var}(\hat{\theta}) \geq 1/g(\theta)$.

The Fisher–Rao metric is the precise object that makes Pillars IV and V one: the "state manifold" of law-state, when its points are evidential distributions, is a statistical manifold whose Riemannian metric (Definition 3.24 in spirit) is Fisher–Rao, and along which the gradient descent of Φ_G (Proposition 3.6) is natural-gradient descent. This is not idle unification: the recent geometry-of- learning literature studies exactly such Riemannian and representation-geometric structure in trained models (David, 2025; Yadav et al., 2026), and the renormalisation and scaling analyses of Section 2.7 live on the same manifolds (Y. Tan et al., 2026; Montanari and Z. Wang, 2026).

3.5.5 The information cost of standardisation

Finally, Pillar III's "latent heat of standardisation" (Definition 4.4) has an information reading. To agree on a protocol is to fix a shared code; specifying that code has a description length in bits, and Landauer's principle prices the thermodynamic cost of irreversible information operations at $k_B T \ln 2$ per bit. The plateau of Fig. 1.1—effort with no visible capability—is, in this reading, the paying-down of the description length of the shared substrate: an information debt incurred once and amortised over every later composition. The three pillars meet here. Standardisation is a phase transition

(III) whose order parameter lives on a statistical manifold (IV/V) and whose latent heat is, to within Landauer's constant, the information content of the agreement. The thesis's metaphor is, at its base, a single statement in three dialects: thermodynamic, geometric, and informational.

3.6 Synthesis: The Conformance Functor

The five pillars converge on a single object. Pillar I gave categories and the monoidal product; Pillar II gave the verdict lattice and monotonicity; Pillars III and IV gave the transition and the manifold; Pillar V gave the measure. We now show that conformance-for-all-language is one *functor*, and that the thesis's three load-bearing results—non-collapse, refusal-survives-composition, and admission-carries-measure—are its three defining properties.

Definition 3.29 (Event category and verdict category)**.** For a domain d (Chapter 8), let the *event category* $\mathcal{E}_d$ have as objects the object-centric event logs over d's schema and as morphisms the *evidence refinements* (log extensions that add events or resolve attributes without contradicting existing order). Let the *verdict category* $\mathcal{L}$ be the verdict lattice of Definition 3.13 regarded as a thin category: objects are verdict assignments and there is a unique morphism $v \to w$ exactly when $v \sqsubseteq w$ in the knowledge order.

Theorem 3.14 (The conformance functor)**.** *Fix for each domain d a model M_d, and let* $\mathrm{Conf}_d \colon \mathcal{E}_d \to \mathcal{L}$ *send a log to its verdict against M_d (Definition 4.3). Then:*

(i) each Conf_d *is a functor;*

(ii) the domains assemble into the coproduct category $\mathcal{E} = \coprod_d \mathcal{E}_d$, and the copairing $\mathrm{Conf} = [\mathrm{Conf}_d]_d \colon \mathcal{E} \to \mathcal{L}$ *is a single functor—one conformance surface for all language;*

(iii) Conf *is monotone (Theorem 3.6): it sends evidence refinements to knowledge increases and never to collapses.*

Proof. (i) A functor must preserve identities and composition. The identity refinement (adding nothing) leaves the verdict fixed, and Conf_d of a composite refinement equals the composite of verdict morphisms because verdict depends only on the accumulated evidence, which composes associatively (Proposition 3.4). Functoriality requires that a refinement $\ell \to \ell'$ map to a morphism $\mathrm{Conf}_d(\ell) \to \mathrm{Conf}_d(\ell')$ in $\mathcal{L}$, i.e. to a knowledge increase $\sqsubseteq$; this is exactly the monotonicity of Theorem 3.6. (ii) The coproduct $\mathcal{E}$ has the universal property that a functor out of it is precisely a family of functors out of the summands; the copairing $[\mathrm{Conf}_d]_d$ is that functor, defined on the summand $\mathcal{E}_d$ as Conf_d. (iii) is (i) read across all summands. □

The thesis's two further results are now properties of this functor.

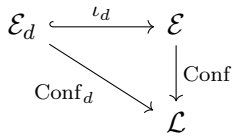

Figure 3.2: Each domain's conformance Conf_d factors through the one functor Conf on the coproduct of event categories (Theorem 3.14). "All language, one surface" is the commutativity of this triangle for every domain d.

Proposition 3.10 (Refusal survives composition)**.** *Equip $\mathcal{E}$ with the monoidal product $\otimes$ that fuses two logs (Pillar I) and $\mathcal{L}$ with the axiswise truth-order meet $\wedge_t$. Then $\mathrm{Conf}(x \otimes y) \preceq_t \mathrm{Conf}(x) \wedge_t \mathrm{Conf}(y)$: on each axis the fused verdict is no more admitted than the meet of the parts. In particular a* REFUSED *axis in either part is* REFUSED *in the fusion.*

Proof. On an axis a, the fused log contains the evidence of both parts. If either part refuses a (witnessing a violation), the violation persists in the fusion, so the fused verdict is $\mathsf{f} = \min_t$. Admission (t) of the fusion requires the violation-free evidence of both, hence both parts admit; and an axis UNKNOWN in either part cannot exceed u in the fusion without new evidence (Theorem 3.6). In every case the fused value is $\leq_t \min_t(\mathrm{Conf}(x)_a, \mathrm{Conf}(y)_a)$. □

Proposition 3.10 is the compositor invariant of Section 7.5 ("`REFUSED_BY_LAW` codes survive always") stated as a lax-monoidality, and it is the formal guarantee behind the portable verdict of Section 7.6: fusing more agents can only lower, never launder, an admission.

Proposition 3.11 (Admission carries measure)**.** *An* ADMITTED *verdict of* Conf *on a trajectory is accompanied by positive receipt measure: $\mu_R(\{\gamma\}) > 0$ (Proposition 3.7). Equivalently,* Conf *factors through the witnessed set W, and admission off W is not in the image.*

Proof. Admission requires a receipt by Definition 4.5; a receipt assigns positive weight $w(\gamma) > 0$, so $\mu_R(\{\gamma\}) > 0$ by Construction 3.3. Trajectories with $w = 0$ lie outside W and cannot be admitted, so Conf restricted to admission has image in the witnessed set. □

Corollary 3.15 (All language, one functor)**.** *The seven domains of Chapter 8—natural language, legal text, mathematics, biological sequence, music, business process, and the reflexive case of an agent's own conduct—are objects of $\mathcal{E}$, and a single functor* Conf *(Theorem 3.14), monotone (Theorem 3.6), lax-monoidal (Proposition 3.10), and measure-valued (Proposition 3.11), is their common conformance surface. Passing from one domain to another changes the object and the model M_d; it does not change the functor.*

Corollary 3.15 is the mathematical statement of the thesis. The reach of the fused protocols (Pillar I) and the trustworthiness of three-valued, receipt-bound, measure-carrying conformance (Pillars II and V) are not two systems bolted together but one functor on one category—and the phase transition of Pillars III and IV is what carries that functor from the single domain where van der Aalst built it to the whole of language.

Chapter 4

Conceptual Framework: Language as Process

> All models are wrong, but some are useful.
>
> George E. P. Box

This chapter supplies the theory the empirical and design chapters depend on. It makes one move and follows its consequences. The move is to treat *any* use of *any* language as behaviour that leaves a trace; the consequences are an event-log abstraction for language (Section 4.1), a phase-transition model of protocol convergence (Section 4.2), a latent-heat account of standardisation (Section 4.3), a definition of the law-state runtime (Section 4.4), and the unified-substrate hypothesis that ties them together (Section 4.5).

4.1 From Text to Event Log: Language as Behaviour

Process mining begins where there is an event log; this thesis begins by observing that language use *is* an event log waiting to be read.

Definition 4.1 (Machine-mediated language)**.** A *machine-mediated language* is any symbolic system—programming language, natural language, legal or regulatory text, mathematical notation, biological sequence, musical score, or business-process notation—whose production, interpretation, or transformation is carried out with, or observed by, a computational agent.

The conventional view treats a text as a static object that a model *reads*. The process view treats the same text as the residue of behaviour: a document was drafted, clauses were inserted and revised, a proof was constructed step by step, a melody was stated and then varied, a sequence was transcribed and edited. Each such act is an *event*, with a timestamp, an actor, and a set of objects it touches.

Definition 4.2 (Event-log abstraction of a language)**.** A language L *admits an event-log abstraction* when its use can be recorded as a set of events, each carrying (i) an activity, (ii) a time order, and (iii) a reference to one or more typed objects, such that the record is an object-centric event log in the sense of OCEL 2.0 (Berti et al., 2024b).

Definition 4.2 is the hinge of the thesis. It converts the open-ended question "can a language server serve domain X?" into the tractable question "does X admit an event-log abstraction?". Where the conventional language server answers queries *about a snapshot of text*, the process view lets a server reason about *the behaviour that produced and will act on that text.*

Definition 4.3 (Conformance surface)**.** Given a language L with an event-log abstraction and a model M_L of intended behaviour, the *conformance surface* of L is the function that maps an observed log to a verdict locating where the log fits M_L, where it deviates, and where the relationship is undetermined—reported with the four quality dimensions of Section 2.1.3.

Principle 4.1 (Three-valued verdicts)**.** A conformance surface returns one of three verdicts per law axis—ADMITTED, REFUSED, or UNKNOWN—and never collapses UNKNOWN into either polarity. Absence of a refutation is not admission.

Principle 4.1 is inherited directly from the artifact (Chapter 6) and is the epistemic backbone of the whole construction. A binary conformance verdict (fit / not-fit) is a category error when applied to open-world language: most of what an agent could check, it has not yet checked. The third value records that gap honestly.

4.2 The Phase-Transition Model

We now make Hypothesis 1.1 precise enough to be argued with. Let the *addressable surface* of machine-mediated language be, informally, the product of three quantities: the number of *domains* treated as servable languages, the number of *capabilities* (completion, diagnosis, conformance, repair, coordination) exposed over each, and the number of *agents* able to invoke them. Write this order parameter Φ. Let the control parameter be σ, the degree of protocol standardisation and adoption across the LSP, MCP, and A2A strata.

Proposition 4.1 (Discontinuous expansion)**.** Φ *is not a smooth function of* σ*. Below a critical adoption threshold* σ_c*, additions to* σ *yield roughly proportional additions to* Φ *(linear integration). At* σ_c*, the decoupling that each protocol performs—*$M \times N \to M+N$*—begins to* compose across layers*, so that a new domain, a new capability, and a new agent each multiply rather than add to reach. In the neighbourhood of* σ_c*,* Φ *expands by orders of magnitude for a bounded change in* σ*: a phase transition.*

The mechanism behind Proposition 4.1 is compositional, not mystical. Under pre-protocol conditions, connecting D domains, C capabilities, and A agents costs on

the order of $D \cdot C \cdot A$ bespoke integrations. Under the fused protocols, each domain is integrated once (an LSP-style server), each capability once (an MCP-style primitive), and each agent once (an A2A-style card), so the cost falls toward $D + C + A$ while the *reachable combinations* remain $D \cdot C \cdot A$. The ratio of reachable surface to integration cost—roughly $\frac{DCA}{D+C+A}$—is the quantity that explodes. This is the same inversion LSP achieved for editors and languages, now compounded over three strata.

Remark. The metaphor's numerical anchor is deliberately physical. Water at the boiling point expands by ∼1,600–1,700× (University of Illinois Department of Physics, 2010; The National Board of Boiler and Pressure Vessel Inspectors, 2010); the claim of Proposition 4.1 is not that the digital expansion equals this figure, but that it is of the *same character*—discontinuous and large—rather than incremental. We use the figure as an order-of-magnitude emblem, and we mark any literal projection of Φ as a FORECAST (Chapter 9).

4.3 Latent Heat and the Energetics of Standardisation

Phase transitions are paid for. The distinctive thermodynamic fact is that, at the transition, energy enters the system *without raising its temperature*: the latent heat of vaporisation is absorbed at constant 100 °C to break intermolecular bonds and reorganise liquid into vapour (OpenStax, 2012). We propose an analogue.

Definition 4.4 (Latent heat of standardisation)**.** The *latent heat of standardisation* is the cumulative effort invested in specifying, ratifying, and adopting a protocol that produces *no visible capability gain while it is being invested*, and that conditions the subsequent discontinuous expansion of the addressable surface.

In the model of Fig. 1.1, "temperature" is visible capability and "energy" is cumulative standardisation effort. During the plateau, d(capability)/d(effort) ≈ 0: years of specification work register as apparent stagnation. The plateau is not waste; it is the precondition. The history of Chapter 2 is legible this way—LSP (2016), MCP (2024), A2A (2025), and their arrival under a shared foundation (2025)—as the staged absorption of latent heat (The Linux Foundation, 2025b; The Linux Foundation, 2025a).

Principle 4.2 (Plateaus are not failures)**.** A period in which standardisation effort yields no visible capability is evidence consistent with an approaching transition, not evidence against it. Evaluations conducted mid-plateau systematically understate eventual Φ.

Principle 4.2 has a sober corollary for Chapter 9: it also licenses over-optimism. A plateau is consistent with an *approaching* transition and with one that never arrives. The principle therefore constrains how forecasts are stated, not whether they are believed.

4.4 Law-State Runtime: Conformance as a First-Class Surface

The protocols move messages; conformance judges behaviour. To judge behaviour during operation, conformance must be a runtime surface, not an after-the-fact report.

Definition 4.5 (Law-state runtime)**.** A *law-state runtime* is a system that (i) observes the behaviour of agents over machine-mediated language as an object-centric event log, (ii) maintains a conformance surface over that log against declared law axes, (iii) exposes the resulting ADMITTED/REFUSED/UNKNOWN state through a protocol so that agents, CI systems, and release gates can read it live, and (iv) admits claims of conformance only when bound to a verifiable *receipt*.

A law-state runtime is to conformance what a language server is to language intelligence: a decoupled, networked surface that any client may consult. Its read-only posture toward the artifacts it judges (Chapter 6) is what keeps it trustworthy: an observer that can silently rewrite what it observes is not an oracle but a participant.

4.5 The Unified-Substrate Hypothesis

The strands now combine.

Hypothesis 4.1 (Unified substrate)**.** *LSP (generalised per Definitions 4.1 and 4.2), MCP, and A2A are not three protocols but three strata of one substrate for machine-mediated language: LSP the* semantic *stratum (what an utterance means), MCP the* contextual *stratum (what an agent may consult to interpret or act), and A2A the* coordinative *stratum (how agents combine). A law-state runtime (Definition 4.5) is the cross-cutting* conformance *stratum that makes the substrate trustworthy. The fusion of all four is the phase change of Hypothesis 1.1.*

Hypothesis 4.1 is evaluated, not asserted: Chapter 7 composes the strata in the artifact, Chapter 8 tests the semantic-plus-conformance strata across six domains, and Chapter 10 grades the hypothesis with bounded status. Figure 4.1 summarises the pipeline the hypothesis implies: an utterance in any language becomes an object-centric log, which a conformance surface maps to a three-valued verdict, which the protocols carry to whichever client needs it.

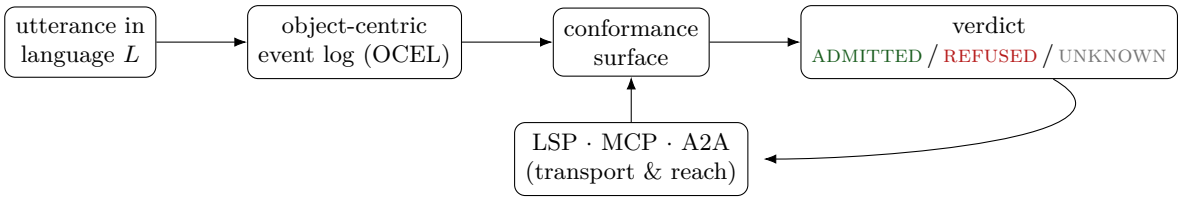

Figure 4.1: The pipeline implied by the unified-substrate hypothesis. Any language admitting an event-log abstraction (Definition 4.2) is mapped by a conformance surface (Definition 4.3) to a three-valued verdict (Principle 4.1); the three protocols carry both the inputs and the verdict to any client.

Chapter 5

Research Methodology

> The artifact and the theory it embodies are evaluated together.
>
> after March and Smith (1995)

5.1 Design Science Research

This thesis is a work of *design science*: it produces and evaluates an artifact in order to learn something general, rather than observing a pre-existing phenomenon. The paradigm is set out by March and Smith (1995), who distinguish *build* and *evaluate* as the core design activities, and by Hevner et al. (2004), whose three-cycle view—a *relevance* cycle to the application environment, a *rigour* cycle to the existing knowledge base, and a central *design* cycle of building and evaluating—structures the work reported here. The design-science research methodology (DSRM) of Peffers et al. (2007) supplies the process spine: problem identification, objectives, design and development, demonstration, evaluation, and communication. Wieringa (2014) frames the alternation of *design problems* and *knowledge questions* that this thesis interleaves.

Table 5.1 maps the DSRM activities onto the chapters, so that the methodology is legible as a thread through the document rather than a detached preamble.

5.2 The Artifact and Its Study

The central artifact is LSP-MAX, a law-state runtime studied as it exists in the repository at CalVer 26.6.18 (lsp-max contributors, 2026). The thesis does not report the construction of a greenfield system; it analyses an extant, non-trivial codebase as the instantiation of Definition 4.5, and reasons from its design. The study proceeded by (i) reading the artifact's constitution and architecture documents, (ii) tracing its protocol surface, conformance types, and enforcement mechanisms to source with file-and-line precision, and (iii) relating each mechanism to the conceptual framework

Table 5.1: The design-science research methodology of Peffers et al. (2007) mapped onto this thesis.

DSRM activity	Realisation in this thesis	Chapter
Problem identification	Narrowness of machine language; reach without accountability; fragmentation	Chapter 1
Objectives of a solution	The unified-substrate and phase-transition hypotheses; RQ1–RQ5	Chapters 1 and 4
Design & development	The LSP-MAX artifact and the fusion architecture	Chapters 6 and 7
Demonstration	Conformance across six language domains	Chapter 8
Evaluation	RQs revisited; bounded-status grading; threats to validity	Chapter 10
Communication	The thesis itself, and the artifact's public surfaces	all

of Chapter 4. Where the artifact references sibling systems that were not present in the study environment—the `wasm4pm` execution engine and its type-authority crate among them—those systems are characterised from the artifact's own interfaces and documentation, and the gap is recorded as UNKNOWN rather than papered over.

5.3 Evaluation Strategy

Design-science artifacts admit several evaluation methods (Hevner et al., 2004); this thesis uses two. The first is *analytical*: the artifact's mechanisms are examined against the properties Chapter 4 requires of a law-state runtime—three-valued verdicts, read-only observation, receipt-bound admission, object-centric logging—and judged present, partial, or absent. The second is *illustrative scenarios*: Chapter 8 works each of six language domains through the pipeline of Fig. 4.1 to show, by construction, that the event-log abstraction and conformance surface apply. Illustrative scenarios demonstrate feasibility and expose limits; they do not establish field efficacy, and Chapter 10 is explicit that broad empirical validation across deployed systems remains OPEN.

5.4 Bounded-Status Epistemics

The artifact under study enforces an epistemic discipline through its constitution and its `anti-llm-cheat-lsp` canary: claims must carry *bounded status*, "victory language" is forbidden, and an assertion of admission requires a *receipt*—a content-addressed record binding a digest, a digest algorithm, a boundary, a checkpoint, the raw command, and, where required, a negative-control result (lsp-max contributors, 2026). This thesis adopts the same discipline reflexively, for two reasons. Methodologically, it is the honest register for design-science claims, which are warranted to varying and stateable degrees. Substantively, it would be incoherent to describe a conformance artifact in the unbounded language that artifact refuses.

Table 5.2: The bounded-status vocabulary used to grade findings, adopted from the artifact's constitution (lsp-max contributors, 2026).

Status	Meaning in this thesis
ADMITTED	Evidence is present and has been checked against the relevant criterion.
REFUSED	Evidence is present and is contrary to the claim.
UNKNOWN	Admissibility cannot yet be determined; a precondition or trace is missing.
CANDIDATE	A design or claim is plausible and partially supported, not yet admitted.
PARTIAL	Admitted in part; a stated remainder is outstanding.
BLOCKED	Determination is prevented by an external precondition.
OPEN	A known gap that no current evidence closes.
FORECAST	A projection about the future; not a measurement.

Concretely, every substantive finding is graded with one of the bounded statuses in Table 5.2, and the three conformance verdicts never collapse (Principle 4.1). Numerical claims about the future are marked FORECAST and attributed to their forecasting body (Chapter 9); quotations from living specifications are pinned to the revision in force.

5.5 Threats to Validity (Preview)

The threats are stated in full in Section 10.2; they are previewed here so that the reader carries them through the design chapters. *Construct validity* is at risk where the water-to-steam metaphor is taken too literally; Section 4.2 confines it to an order-of-magnitude, same-character claim. *Internal validity* is at risk where the artifact's documented status is taken for its runtime behaviour; the study relies on source and specification, not on a live deployment exercised across all domains, and says so. *External validity* is at risk in generalising from one artifact and six illustrative domains to "all language"; the event-log criterion (Definition 4.2) is the explicit boundary of the claim. *Reliability* is supported by tracing claims to primary sources and to file-and-line locations, so that a later reader can re-examine them.

Chapter 6

The lsp-max Artifact

> The repository is not passive text. It is a law-bearing system.
>
> — *LSP-MAX*, `AGENTS.md`

This chapter presents LSP-MAX as the instantiation of the law-state runtime of Definition 4.5. It is, in the words of its own manifest, "a law-state runtime projected through LSP" (lsp-max contributors, 2026), and it is studied here at CalVer 26.6.18. The chapter traces its five-layer architecture (Section 6.2), its `max/*` protocol surface (Section 6.3), its three-valued `ConformanceVector` (Section 6.4), its receipts and typestate kernel (Section 6.5), its self-enforcing constitution (Section 6.6), and—the point of contact with van der Aalst's formalisms—its binding of diagnostics to object-centric events (Section 6.7).

6.1 Overview: Inverted LSP

A conventional language server helps a human write code. LSP-MAX inverts the direction: it "makes the repository speak back to agents while they work" (lsp-max contributors, 2026). The same JSON-RPC surface that an editor uses to ask "what does this symbol mean?" is repurposed so that an agent may ask "is this trajectory admissible?". The artifact's constitution, `AGENTS.md`, states the governing condition for admission in a form worth reproducing, because it is Definition 4.5 in the artifact's own notation:

$$\mathrm{Done}_B(A) \;=\; \big[\,\mathrm{FailSet}_B(A) = \varnothing\,\big] \;\wedge\; \big[\,R_B \vdash A = \mu(O^*_B)\,\big].$$

In words: agent output is admitted not because it compiles, logs success, or passes a test, but only when bounded receipts prove the action equals the lawful transformation of admitted observations (lsp-max contributors, 2026). The artifact thus refuses exactly the inference—"the test passed, therefore the work is admissible"—that Principle 4.1 forbids.

Table 6.1: The five-layer model and its crate mapping (lsp-max contributors, 2026). Layer numbering runs from the actuation grammar (1) up to the autonomic mesh (5); "law-state" (Layer 3) is the load-bearing abstraction.

Layer	Name	Realising crate(s)
5	Autonomic LSP mesh	`lsp-max-compositor`; example servers
4	Knowledge hooks	`src/composition/`; hook graph
3	Law-state runtime	`lsp-max-runtime`, `lsp-max-protocol`
2	Local LSP state surface	root crate `src/` (`LanguageServer`, `LspService`, `Server`)
1	Actuation grammar	`crates/lsp-max-cli` (clap-noun-verb)

6.2 The Five-Layer Model

LSP-MAX is organised as five layers in which "law-state" is the load-bearing abstraction (`docs/ARCHITECTURE.md`). Table 6.1 lists them with the crates that realise each.

Two design rules of the model matter for the thesis. First, the LSP surface is *read-only*: it may emit diagnostics, hovers, code-action intents, virtual documents, and protocol traces, but "it must not directly mutate files"; any future mutation must route through an explicit chain ending in a receipt (lsp-max contributors, 2026). This realises the read-only posture that Section 4.4 requires of a trustworthy observer. Second, the actuation grammar at Layer 1 is a noun/verb CLI built on `clap-noun-verb`, in which a filename is a noun and an annotated function is a verb—a small but real demonstration that even the command surface is treated as a language with a grammar.

6.3 The `max/*` Protocol Surface

Atop the standard LSP methods, LSP-MAX declares a custom `max/*` method family (`lsp-max-protocol/src/custom_methods.rs`) that exposes the law-state runtime over JSON-RPC. Table 6.2 groups the most load-bearing of these. The surface is the concrete form of Definition 4.5(iii): the conformance state is not an internal detail but a queryable protocol.

Note in particular `max/releaseActuation`: a release is gated on the conformance vector admitting it, and `max/refusal` carries a rationale *and* a receipt. Admission and refusal are first-class, symmetric, and evidence-bound.

6.4 The `ConformanceVector` and Three-Valued Law Logic

The artifact's core type is the realisation of Principle 4.1. It keeps three disjoint sets of law axes and never coerces one into another (`lsp-max-protocol/src/conformance.rs`).

```
pub struct ConformanceVector {
    pub admitted: Vec<LawAxis>,  // evidence present and valid
    pub refused:  Vec<LawAxis>,  // evidence present, violation
        confirmed
```

Table 6.2: Representative methods of the `max/*` surface (lsp-max contributors, 2026). The family exposes snapshots, three-valued admission, receipts, repair plans, and release actuation over JSON-RPC.

Method	Role
`max/snapshot`, `max/manifoldSnapshot`	capture server / full law-state snapshot
`max/conformanceVector`	query the three-valued conformance state at a snapshot
`max/admission`	admissibility gate over a law axis (ADMITTED/REFUSED/UNKNOWN)
`max/refusal`	explicit refusal with rationale and receipt
`max/lawfulTransition`	assert that a phase transition is lawful
`max/explainDiagnostic`, `max/repairPlan`	explain a diagnostic; emit read-only repair steps
`max/receipt`, `max/replay`	retrieve a receipt; replay the event log against a snapshot
`max/releaseActuation`	actuate a release *iff* the `ConformanceVector` admits it
`max/workspaceConformance`, `max/lsif`	aggregate workspace conformance; stream an LSIF graph

```
    pub unknown:  Vec<LawAxis>,  // admissibility NOT determinable
    pub score: Option<f64>,      // 100 * admitted / (admitted+
        refused+unknown)
    pub strict_mode: bool,       // do unknown axes block release?
    pub process_quality: Option<ConformanceResult>, // POWL
        conformance
    // bitmasks (not serialized) enforce mutual disjointness
}

impl ConformanceVector {
    pub fn all_admitted(&self) -> bool {
        self.refused.is_empty() && self.unknown.is_empty()
    }
    pub fn admits_release(&self) -> bool {
        self.refused.is_empty() && (!self.strict_mode || self.unknown
            .is_empty())
    }
}
```

Listing 6.1: The three-valued conformance state (abridged from `conformance.rs`).

Three properties deserve comment. First, UNKNOWN is a peer of ADMITTED and REFUSED, not the absence of a verdict; the three backing bitmasks are mutually disjoint and an invariant check asserts it. Second, `admits_release` encodes a genuine policy choice: in `strict_mode`, unknown axes block release, so that "not yet checked" is treated as "not yet permitted". Third, the law axes are an enumeration—`Protocol`, `Type`, `Fixture`, `Documentation`, `Release`, `Hook`, `Repair`, `Receipt`, `Security`, `Autopoiesis`, `Domain`, and an extensible `Custom(String)`—so the same machinery scales from protocol conformance to arbitrary domain law, which is exactly the extensibility Chapter 8 will exploit. The `process_quality` field, carrying a conformance result from the

process-mining engine, is the seam through which van der Aalst's machinery enters the type.

6.5 Receipts, Snapshots, and the Typestate Kernel

A claim of admission is worthless without something that cannot be cheaply forged. LSP-MAX binds admission to a *receipt* whose hash chains to its predecessor (`lsp-max-protocol/src/core.`

```rust
pub struct Receipt {
    pub receipt_id: String,
    pub hash: String,                          // SHA-256 digest of the
        artifact
    pub prev_receipt_hash: Option<String>,   // None only for the
        genesis receipt
}
```

Listing 6.2: Content-addressed receipts form a hash chain (abridged from `core.rs`).

Each receipt's `prev_receipt_hash` points to the digest of the previous receipt, forming a tamper-evident ledger whose genesis link is the only one permitted to be `None`. The compositor additionally issues a `CryptographicReceipt` chain (BLAKE3 hashing with Ed25519 signatures) so that per-server contributions to a merged verdict remain attributable (lsp-max contributors, 2026). A companion script validates the chain, requiring boundary markers (`---BEGIN RECEIPT---`), a 64-hex digest, and a bounded status word, and returning REFUSED when any are absent—so that the validator itself obeys the three-valued discipline.

State evolution is governed by a typestate kernel (`lsp-max-runtime/src/typestate/`) whose phases are monotonic—`Uninitialized` → `Initializing` → `Initialized` → `ShutDown` → `Exited`, with no back-edges—and whose trait exposes a `validate` → `select` → `admit` → `receipt` → `exit` → `replay` life cycle. The presence of `replay` is significant: the runtime can reconstruct its state from the receipt ledger, so the law-state is not merely asserted but *re-derivable* from evidence—the computational form of conformance checking's "replay the log against the model".

6.6 The Anti-Cheat Canary and the Enforceable Constitution

What makes LSP-MAX a *law*-state runtime rather than a state machine with aspirational documentation is that its constitution is enforced by a component of the system itself: `anti-llm-cheat-lsp`, the "diagnostic canary". It runs on LSP-MAX, exercises the LSP surface, and detects attempts to reintroduce the prohibited base framework that LSP-MAX forks, to forge receipts or routes, or to use the language of unbounded success (lsp-max contributors, 2026). The self-sealing property is stated plainly in

the constitution: LSP-MAX hosts the canary; the canary detects regression; therefore LSP-MAX cannot silently regress (lsp-max contributors, 2026).

Detection is layered—raw-text scan, tree-sitter AST scan, manifest and dependency-graph scan, claim scan over Markdown and agent reports, JSON-RPC transcript scan, a receipt validator, a route-evidence checker, and a claim-versus-proof checker—and every diagnostic must name the *forbidden implication* it prevents (for example, "TestStdout ⇒ Receipt") (lsp-max contributors, 2026). The diagnostic families span surface, authority, receipt, route, mutation, test, version, and overclaim violations. Two enforcement mechanisms are directly relevant to the thesis.

Bounded status and the refusal of victory language. The constitution forbids the words of triumph and enumerates the only permitted statuses—the same vocabulary this thesis adopts in Table 5.2. The canary's overclaim family raises a diagnostic when forbidden terms appear, using an Aho–Corasick automaton for linear-time classification (lsp-max contributors, 2026).

The ANDON gate. A pre-tool-use hook runs `lsp-max-cli gate check` before each shell, edit, or write action; exit 1 blocks the action while an ANDON signal is set. The governing predicate is

$$\Lambda_{CD}(a) \;=\; \Lambda(a) \;\wedge\; \neg\,\exists\, d \in D_t : d.law\_id \in \mathcal{A} \;\wedge\; d.severity = \textsc{Error},$$

where $\Lambda(a)$ is the base admissibility predicate, D_t the live diagnostic set, and $\mathcal{A}$ the governed law axes (`WASM4PM-*`, `ANTI-LLM-*`, `GGEN-*`) (lsp-max contributors, 2026). The gate state is published as a single byte in a deterministically named file, so any process—including a subagent—can read it in one syscall. The artifact is candid that subagent hooks do not cross session boundaries, recording this as OPEN rather than ADMITTED: an honesty the thesis takes as a model.

6.7 OCEL Binding: Diagnostics as Object-Centric Events

The artifact's deepest point of contact with the paradigm of Section 2.1—and the strongest single piece of evidence for Hypothesis 4.1—is that it does not merely *borrow vocabulary* from process mining; it *runs* on it.

First, the constitution's laws are expressed as *Declare* constraints—the declarative, LTL-based process-modelling formalism from van der Aalst's school—and mapped to diagnostics (`declare_laws.rs`): the prohibition on the base framework becomes `Absence`; "a diagnostic must produce an OCEL event" becomes `Response`; "a receipt must precede an admission" becomes `Precedence`; "test output and a receipt cannot co-occur" becomes `NotCoexistence`; and "an unknown axis cannot be succeeded by an admitted one without a resolution event" becomes `NotSuccession`. The artifact's

Table 6.3: The `wasm4pm` Oracle classes recast cheating as object-centric axiom violations (lsp-max contributors, 2026). The last row is the computational guardian of Principle 4.1.

Oracle	**Formalised violation**
A8 Audit-log tampering	an event with no causal predecessor
A9 Temporal anomaly	$e_1 \to e_2$ in causality but $t(e_1) \geq t(e_2)$
A10 Causal violation	a mutation with no preceding observation
A11 Unknown-state collapse	UNKNOWN$\to$ {ADMITTED, REFUSED} without a resolution event
A12 Cyclic dependency	a cycle in the causality graph

governance is, literally, a declarative process model checked for conformance.

Second, each diagnostic emission is recorded as an object-centric event (`ocel.rs`). The object types are `CaseFile`, `DetectionCode`, `LawAxis`, `Receipt`, `Gate`, and `CodeSymbol`; a `CheatDetected` event binds a file, a detection code, and a law axis, and a `ScanComplete` event closes over the files scanned. The result is an OCEL log of the agent's own conduct that any process-mining tool can read.

Third, conformance of that log is checked by the `wasm4pm` engine, which returns a `GallVerdict`—`Fit{fitness}`, `Deviation{fitness, missing}`, `Blocked{reason}`, or `Inconclusive{reason}`—whose four cases map cleanly onto the bounded statuses and, notably, reserve `Inconclusive` for the UNKNOWN case. The engine's type-authority crate further encodes OCEL axioms (disjoint event and object universes; mandatory event-to-object cardinality; temporal continuity; type closure) and a family of *Oracle classes* (A8–A12) that recast adversarial behaviour—audit-log tampering, temporal anomaly, causal violation, unknown-state collapse, cyclic dependency—as *violations of those axioms* (lsp-max contributors, 2026). Anti-cheat detection is thereby lifted from pattern-matching to specification-derived conformance checking.

The artifact therefore exhibits, in miniature, the entire claim of this thesis: a non-code "language"—the agent's own stream of edits, claims, and commands—is given an event-log abstraction (Definition 4.2), checked against a declarative model on a conformance surface (Definition 4.3), and reported in three-valued, receipt-bound, read-only terms. What Chapter 8 does is widen the aperture from the agent's conduct to natural language, law, mathematics, biology, music, and business process. The mechanism is already present; only the domain changes.

Chapter 7

The Fusion Architecture: LSP × MCP × A2A

> A single Language Server can be re-used in multiple development tools, which in turn can support multiple languages.
>
> *Microsoft*, on LSP (Microsoft, 2024b)

Chapter 2 established that LSP, MCP, and A2A are the same decoupling at three layers; Chapter 4 hypothesised that they are therefore strata of one substrate (Hypothesis 4.1); Chapter 6 showed an artifact that already runs the conformance stratum. This chapter composes the strata explicitly. It assigns each protocol its role (Sections 7.2 to 7.4), shows how LSP-MAX occupies more than one stratum at once (Section 7.5), and traces a single conformance verdict through the whole stack (Section 7.6).

7.1 The Convergence Thesis, Concretely

The fusion claim is not that the three protocols will be merged into one specification. It is that they *compose*: because all three are JSON-RPC contracts with explicit capability negotiation (Microsoft, 2024a; Model Context Protocol, 2025b; A2A Project, 2025c), a single agent can speak all three without impedance, and the output of one becomes the input of another. An agent discovered over A2A consults tools over MCP and reasons about language over a generalised LSP; nothing in the three specifications obstructs this layering, and their shared lineage and governance actively invite it (Model Context Protocol, 2025b; A2A Project, 2025a; The Linux Foundation, 2025a). Figure 7.1 renders the resulting architecture.

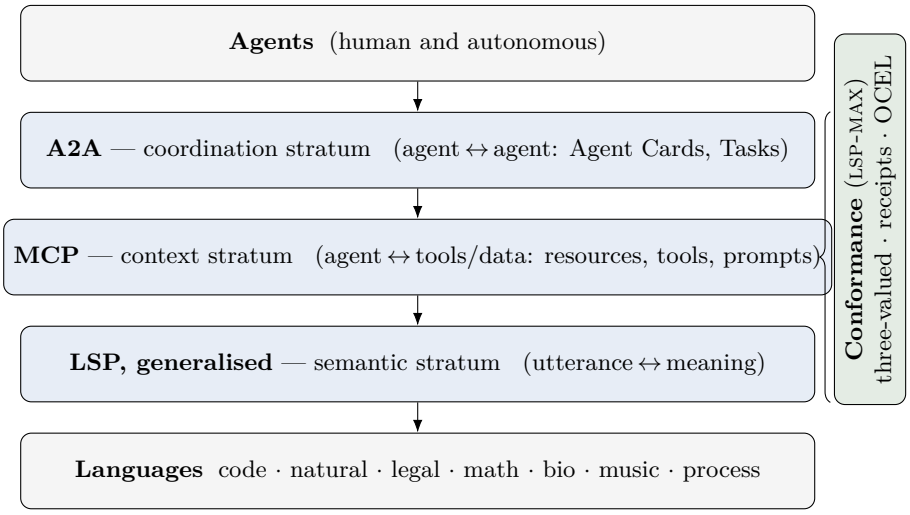

Figure 7.1: The fused substrate. A2A, MCP, and a generalised LSP form three horizontal strata between agents and languages; LSP-MAX supplies a cross-cutting conformance stratum (Hypothesis 4.1) that observes every layer as an object-centric event log and reports three-valued, receipt-bound verdicts.

7.2 LSP as the Semantic Substrate

The lowest protocol stratum answers "what does this utterance mean?". Its contribution to the fusion is the very property that made it successful for code: it isolates language understanding behind a uniform contract so that the layers above need not know how meaning is computed. Generalised per Definition 4.1, an LSP server need not serve a programming language at all—LTeX already serves natural language, and the YAML server serves a data format (Valentin and contributors, 2023; Red Hat Developer, 2024)—and LSP-MAX extends the surface with the `max/*` family so that "meaning" includes "law-state" (Section 6.3). In the fused architecture, every language domain of Chapter 8 appears as an LSP-style server, integrated once and reachable by every agent above.

7.3 MCP as the Context Substrate

The middle stratum answers "what may I consult, and what may I do?". MCP's three server primitives—resources, prompts, and tools (Model Context Protocol, 2025b)—are the natural envelope in which a generalised LSP server is presented to a model: a language server's analyses become MCP *resources*, its repair plans become *prompts*, and its actuations become *tools*. The composition is direct and already idiomatic, since MCP was itself shaped by LSP (Model Context Protocol, 2025b). Concretely, LSP-MAX's read-only analyses (`max/conformanceVector`, `max/explainDiagnostic`) map to MCP resources, while its guarded actuations (`max/releaseActuation`) map to MCP tools whose invocation the law-state runtime may refuse. MCP thus carries

conformance *outward* to any host application without that host needing to understand the conformance machinery.

7.4 A2A as the Coordination Substrate

The upper stratum answers "which agents exist, and how do we combine?". A2A's Agent Card advertises capabilities for discovery (A2A Project, 2025c; A2A Project, 2025b); in the fused architecture, "produces receipt-bound conformance verdicts" is itself an advertisable capability, so that a coordinating agent can locate a *conformance authority* the way it locates any other skill. A2A's Task lifecycle—`submitted`, `working`, `input-required`, `completed`, `failed`—is, read through Chapter 4, already an event log: each state transition is an event with a timestamp and an actor. The coordination stratum thus not only *carries* conformance verdicts between agents; its own activity is a language with a conformance surface, exactly as Section 6.7 showed for an agent's stream of edits.

7.5 The Composed Stack: lsp-max as Mesh

LSP-MAX is instructive because it does not sit in a single stratum. Its Layer-2 surface is an LSP server (semantics); its `max/*` methods and `ConformanceVector` are the conformance stratum; and its Layer-5 *compositor* is a multi-server mesh that fans a client request out to N child servers and merges their diagnostics, with the rule that refusals survive the merge ("`REFUSED_BY_LAW` codes survive always") (lsp-max contributors, 2026). This last property is the architectural expression of Principle 4.1 at the mesh level: aggregating many servers must not let an admission elsewhere overwrite a refusal here. The compositor emits its own merge receipts and can render the fan-out-then-merge flush as an OCEL event, so that even the act of composition is minable. A mesh of conformance authorities, coordinated over A2A and consulted over MCP, each understanding some language over LSP, is the fused substrate made operational.

7.6 Conformance Across the Fusion

The point of fusing the strata is that a single verdict can travel the whole height of Fig. 7.1. Consider a coordinating agent that must release a regulated artifact. It discovers, over A2A, a conformance authority whose Agent Card advertises the capability. It invokes that authority's analysis as an MCP tool. The authority is, internally, an LSP-MAX mesh: it treats the artifact as an object-centric event log (Definition 4.2), runs the conformance surface over the relevant law axes (Definition 4.3), and returns a `ConformanceVector`. If any axis is REFUSED, `max/releaseActuation` refuses and emits a refusal receipt; if axes are UNKNOWN and the policy is strict, release is withheld

rather than granted; only if the vector admits release does the coordinating agent proceed—carrying the receipt as evidence that other agents, over A2A, can in turn verify.

Two features of this flow are worth isolating as the fusion's distinctive contributions. First, *accountability composes with reach*: the same capability-negotiated, JSON-RPC machinery that lets the agent reach further also lets a verdict and its receipt travel back, so expansion of reach need not outrun expansion of accountability—the deficiency P2 of Section 1.2. Second, *the verdict is portable*: because the receipt is content-addressed and chained (Listing 6.2), an admission produced deep in an LSP-MAX mesh remains checkable after it has crossed an MCP tool boundary and an A2A task boundary. The fusion does not merely connect systems; it lets a single, three-valued, evidence-bound judgement remain meaningful across every connection it crosses. Whether this portability holds under adversarial conditions at scale is, properly, CANDIDATE rather than ADMITTED, and Chapter 10 returns to it.

Chapter 8

Conformance Checking for All Language

> In the case of LTeX, the "language" is not a single natural language, but all natural languages.
>
> *LTeX documentation* (Valentin and contributors, 2023)

This chapter discharges contributions C2 and C5. It states the criterion that makes a domain a servable language (Section 8.1), then works six domains through the pipeline of Fig. 4.1—natural and legal language (Sections 8.2 and 8.3), mathematics (Section 8.4), biological sequence (Section 8.5), music (Section 8.6), and business process (Section 8.7)—adds the reflexive case of an agent's own conduct (Section 8.8), and closes with the object-centric bridge and the single conformance functor that unify them (Sections 3.6 and 8.9). Per Section 5.3, these are illustrative scenarios: they demonstrate feasibility by construction and expose limits; they are graded CANDIDATE, not ADMITTED, and field validation is OPEN.

8.1 The Criterion: When Is a Domain a Language?

Definition 4.2 gives the test. A domain is a servable language when its use admits an object-centric event-log abstraction: its activities can be recorded as timed events referring to typed objects. Given that, Definition 4.3 supplies a conformance surface, and the artifact of Chapter 6 supplies the machinery—three-valued verdicts, receipts, OCEL events—without modification, because the law axes are extensible (the `Custom` variant of Listing 6.1). The criterion is deliberately restrictive: a domain whose use cannot be observed as events (a purely private, untraced mental act) falls outside the claim, and saying so is what keeps "all language" from meaning "everything". Table 8.1 previews the six demonstrations.

8.2 Natural Language

Natural language is the domain where the semantic stratum is most mature outside code. LTeX serves grammar and style over LSP for "all natural languages supported by LanguageTool" (Valentin and contributors, 2023); textLSP and similar servers add prose checks (Hangya and contributors, 2024). The process view adds what these lack: a memory of behaviour. The *events* are drafting and revision acts; the *objects* are sentences, terms, and claims; a model M_L encodes constraints such as "a defined term is used consistently" or "a claim asserted is not later contradicted". Conformance checking then locates contradiction or terminological drift as a REFUSED axis, with the offending events as the diagnostic's witness, and marks as UNKNOWN those claims whose truth it cannot assess—refusing, per Principle 4.1, to label the unchecked as correct. This is the direct generalisation of the artifact's own claim-versus-proof checker (Section 6.6) from agent reports to arbitrary prose.

8.3 Legal and Regulatory Text

Law is language whose entire purpose is conformance, which makes it the domain where the thesis bites hardest—and the one the artifact's "affidavit" lineage (Chapter 6) already gestures at, through receipts that function as cryptographic attestations. The *events* are the drafting, amendment, and citation of provisions; the *objects* are clauses, obligations, parties, and referenced statutes, with exactly the object-to-object relationships OCEL 2.0 was extended to express (an obligation *borne by* a party, a clause *amending* another) (Berti et al., 2024b). A model M_L encodes obligation consistency (no clause both requires and forbids the same act), citation validity (every reference resolves to an in-force provision), and temporal coherence (amendments respect effective dates—precisely the A9 "temporal anomaly" axiom of Table 6.3). A conformance verdict over a contract or regulation is then a receipt-bound, three-valued document: ADMITTED obligations, REFUSED contradictions, and UNKNOWN provisions whose compliance depends on facts not in evidence. The receipt is the natural digital analogue of a sworn attestation.

8.4 Mathematics and Formal Proof

Mathematical proof is language with the strictest conformance model of all: logical validity. Proof assistants already expose their language over LSP, so the semantic stratum exists. The *events* are the statement of propositions and the application of inference steps; the *objects* are propositions, lemmas, definitions, and terms, related by *depends-on*. The model M_L is the proof calculus itself, and conformance checking is proof checking: a step that does not follow is REFUSED; a lemma invoked before it is established is an A10 "causal violation" (a mutation—use—preceding its observation—

proof); a circular dependency among lemmas is the A12 "cyclic dependency" of Table 6.3. That the artifact's Oracle classes, designed to catch agent cheating, map without strain onto proof pathologies is evidence for the unity the thesis claims.

8.5 Biological Sequence

A genome is a language over a four-letter alphabet with rich grammar. The *events* are transcription, editing, and annotation operations; the *objects* are bases, codons, genes, variants, and regions; the model M_L encodes reading-frame integrity, conformance to a reference sequence, and constraints on permissible edits. Conformance checking flags a frame-shifting edit as REFUSED, a variant of unknown significance as UNKNOWN—a domain that has independently institutionalised Principle 4.1 under the very name "variant of uncertain significance"—and a reference-matching region as ADMITTED. The semantic-stratum tooling here is bioinformatics validation rather than an LSP server today; the contribution of the thesis is to observe that the *same* conformance surface serves it, and that an LSP-style server is a natural packaging.

8.6 Music and Multimodal Notation

A score is notation with grammar (key, metre, voice-leading). The *events* are the statement, transposition, and variation of material; the *objects* are notes, motifs, voices, and bars; the model M_L encodes voice-leading rules, key conformance, and metric consistency. Conformance checking locates parallel fifths or a metric overflow as REFUSED, and leaves stylistically open choices UNKNOWN. Music is included not for its economic weight but because it stresses the criterion of Section 8.1: it shows the event-log abstraction applies to a non-textual, time-structured, aesthetic language, and that "law" need not mean "regulation" to be conformance-checkable.

8.7 Business Process: The Return Home

The sixth domain is the one process mining began with, and including it closes the loop. Here the *events* are activity executions and hand-offs; the *objects* are the multiple case objects whose entanglement motivated OCEL in the first place (Section 2.1.4); the model M_L is a Petri net or a POWL model; and conformance checking is conformance checking in van der Aalst's original sense, computed by the `wasm4pm` engine and returned as a `GallVerdict` (Section 6.7). The thesis adds nothing to this domain that the field did not already have—which is the point. Business process is the *calibration* case: because the artifact's machinery reproduces standard object-centric conformance here, its application to the other five domains is a generalisation of a working method, not an analogy to a foreign one.

8.8 The Reflexive Case: An Agent's Own Conduct

The seventh language is the one the artifact already serves: an agent's own stream of edits, claims, and commands (Section 6.7). The *events* are tool calls, edits, and assertions; the *objects* are the artifact's OCEL types (`CaseFile`, `DetectionCode`, `LawAxis`, `Receipt`, `Gate`, `CodeSymbol`); the model M is the Declare-encoded constitution; and conformance is the canary's verdict, emitted as `CheatDetected` events and checked by the engine's `GallVerdict` (Section 6.7). Two features distinguish this case. First, where business process (Section 8.7) calibrates against van der Aalst's original setting, the reflexive case calibrates against an actually-running implementation, so its status is ADMITTED-by-dogfood rather than CANDIDATE. Second, it is self-applying: the conformance surface is policed by the same machinery it governs, the self-sealing property of Section 6.6. The reflexive case is therefore both the seventh demonstration and the existence proof that the other six are not merely conceivable—the functor of Theorem 3.14 is inhabited on at least one object, witnessed in code.

8.9 The Object-Centric Bridge

What lets one surface serve six domains is the object-centric event log. A flat, single-case log would force each domain into a single "case" notion—one sentence, one proof, one gene—and suffer the convergence and divergence pathologies of Section 2.1.4: a legal obligation borne by several parties, a lemma used in several proofs, a motif recurring in several voices would each be mismodelled. OCEL 2.0's object-to-object and qualified relationships are exactly what these domains require (Berti et al., 2024b). This is the precise sense in which van der Aalst's "no AI without PI" (Aalst, 2025b) becomes, for this thesis, *no conformance for all language without an object-centric abstraction*: the bridge that carries conformance from business process to all language is the object-centric event log, and it is load-bearing. Formally, this unity is Corollary 3.15: the seven domains are objects of one category and conformance is one functor (Theorem 3.14). The unity is summarised by a single observation: across all seven domains, the artifact's machinery changed not at all; only the law axes and the model M_L changed. That invariance, demonstrated by construction, is the evidence offered for Hypothesis 4.1's semantic-plus-conformance claim—offered, per Section 5.3, as CANDIDATE.

Table 8.1: Six domains as servable languages. Each admits an event-log abstraction (activity over typed objects) and therefore a conformance surface; existing tooling shows the semantic stratum is already partly built.

Domain	Event (activity)	Typed objects	Example law axis	Semantic-stratum tooling
Code	edit, parse, build	symbol, module, file	type safety, API stability	LSP servers (Microsoft, 2024b)
Natural language	draft, revise, translate	sentence, term, claim	grammar, style, consistency	LaTeX, textLSP (Valentin and contributors, 2023; Hangya and contributors, 2024)
Legal / regulatory	draft clause, amend, cite	clause, obligation, party	defining consistency, citation validity	
Mathematics	state, step, justify	proposition, lemma, term	logical validity, acyclic dependency	proof assistants
Biological seq.	transcribe, edit, annotate	base, gene, variant	reading frame integrity, reference conformance	bioinformatics validators
Music	state, transpose, vary	note, motif, voice	voice-leading, key/metre conformance	notation formats
Business process	execute, hand off	case objects, resource	control-flow conformance	process mining (Aalst, 2016)

Chapter 9

Vision 2030: The Phase Transition Realised

> Prediction is very difficult, especially about the future.
>
> attributed to Niels Bohr

This chapter discharges contribution C6: a 2030 outlook. It is the most speculative chapter, and it is marked accordingly. Forecasts from analysts and market researchers are reproduced as FORECAST; the thesis's own extrapolations are derived from the model of Chapters 3 and 4 and are likewise bounded. Section 9.1 reviews external forecasts; Section 9.2 applies the phase-transition model; Section 9.3 gives a staged roadmap; and Section 9.4 treats governance and risk.

9.1 External Forecasts

The analyst consensus for the late 2020s is one of rapid agentic adoption paired with high failure rates. Gartner projects that by 2028 one-third of enterprise software applications will include agentic AI, up from less than one per cent in 2024, and that at least fifteen per cent of day-to-day work decisions will be made autonomously (Gartner, 2025c) [FORECAST]; that forty per cent of enterprise applications will feature task-specific agents by 2026 (Gartner, 2025a) [FORECAST]; and, in the same breath, that over forty per cent of agentic-AI projects will be cancelled by the end of 2027, citing "inadequate risk controls" among the causes (Gartner, 2025c) [FORECAST]. Most consequentially for this thesis, Gartner forecasts that "guardian agent" technologies will capture ten to fifteen per cent of the agentic-AI market by 2030 (Gartner, 2025b) [FORECAST]. Market sizings vary widely by definition: one estimate puts the AI-agents market at USD 7.84 billion in 2025 rising to USD 52.62 billion by 2030 (CAGR ≈46%) (MarketsandMarkets, 2025) [FORECAST], another the enterprise segment at USD 2.58 billion rising to USD 24.5 billion (Grand View Research, 2025) [FORECAST]. The spread (a factor of two at the 2030 horizon) is itself the finding: these are indicative ranges, not

measurements.

Two of these forecasts bear directly on the argument. The "inadequate risk controls" cancellation driver is precisely the deficiency P2 of Section 1.2; and the "guardian agent" category is, in this thesis's vocabulary, the market's name for a law-state runtime. The external forecast that a distinct stratum of oversight agents will command a tenth of the market by 2030 is independent, if speculative, support for Hypothesis 4.1's claim that conformance is a stratum in its own right.

9.2 The Expansion Argument

What does the phase-transition model predict, taken seriously and stated honestly? The Decoupling Theorem (Theorem 3.3) and its corollary (Corollary 3.4) establish that, once domains, capabilities, and agents are each integrated once against the fused substrate, the reachable surface scales as the product $\Theta(n^3)$ against linear integration cost $\Theta(n)$. The Landau analysis (Proposition 3.5) establishes that, with a network-effect term, the adoption order parameter jumps discontinuously as standardisation crosses a threshold. Composing the two: *if* adoption crosses its threshold in the late 2020s—of which the institutional consolidation under the Linux Foundation and the Agentic AI Foundation (Section 9.4) is a leading indicator—then the addressable surface of machine-mediated language expands not incrementally but by orders of magnitude, the digital counterpart of the ∼1,600–1,700× expansion that names this thesis.

This is the steam claim, and three honesty constraints bound it. First, it is a FORECAST: the model explains a mechanism, it does not measure Φ. Second, by Principle 4.2, the present period of intense standardisation effort with modest visible capability is consistent with an *approaching* transition and with one that never arrives; the same plateau precedes both. Third, the order-of-magnitude figure is borrowed from physics as an emblem of *character* (discontinuous, large), not as a measured digital quantity. With those constraints, the prediction is: the convergence of LSP, MCP, and A2A under a conformance stratum is of the order type of a first-order phase transition, and the late 2020s are when the threshold is most plausibly crossed.

9.3 A Roadmap to 2030

Table 9.1 stages the transition with bounded entry criteria, so that the roadmap is falsifiable rather than aspirational: each stage names what would count as having entered it.

Stage S1 is already underway: language servers for natural language and data formats exist (Valentin and contributors, 2023; Red Hat Developer, 2024), and the artifact of Chapter 6 emits its own agent conduct as OCEL. Stage S2 is foreshadowed by the workflow-engine and capability-binding preprints of Section 2.7 (Parmar, 2026; Z. Zhou, 2026). Stage S3 awaits A2A-advertised conformance capabilities, of which

Table 9.1: A staged roadmap. Each stage's status is the bounded status of its entry criterion as of submission; later stages are OPEN by construction.

Stage	Entry criterion	Status
S1 (now)	Generalised LSP servers exist for non-code languages; agent traces emitted as OCEL	CANDIDATE
S2	Read-only conformance exposed as MCP tools/resources; receipts cross tool boundaries	CANDIDATE
S3	Conformance authorities discoverable over A2A; verdicts portable across agents	OPEN
S4	Cross-domain law axes standardised under a neutral foundation	OPEN
S5 (2030)	Conformance-for-all-language is a default stratum of the substrate	OPEN

the identity-and-delegation work across MCP and A2A is a precursor (Prakash, 2026). Stages S4 and S5 are OPEN: they require cross-domain law standardisation that does not yet exist, and the thesis does not claim it will.

9.4 Governance, Standardisation, and Risk

The latent heat of standardisation (Definition 4.4) is being paid through institutions. A2A entered the Linux Foundation in June 2025 (Google, 2025; The Linux Foundation, 2025b); MCP entered the Agentic AI Foundation under the same umbrella in December 2025 (Anthropic, 2025; The Linux Foundation, 2025a). Neutral governance is the social mechanism by which a protocol's adoption threshold is crossed, and its arrival for two of the three strata within six months is the clearest leading indicator that the transition's precondition is being met.

The risks are equally concrete, and conformance speaks to several. The cancellation forecast attributes failure partly to weak risk controls (Gartner, 2025c); the security literature documents prompt injection and tool poisoning against MCP (Gaire et al., 2025; C. Huang, X. Huang, et al., 2026; Maloyan and Namiot, 2026); the reliability literature finds that accuracy masks unreliability (Rabanser et al., 2026); and the oversight literature shows that human oversight has finite capacity, so that adding escalation can *reduce* safety past a point (Turan, 2026). A law-state runtime is a partial answer to the first three—a standing, three-valued, receipt-bound risk control—and the "guardian agent" market (Gartner, 2025b) is where it would live. But it is no answer to the fourth: an oversight stratum that itself demands human attention inherits the capacity limit, and "proof-of-guardrail" can be trusted only as far as its own analysis allows (Jin et al., 2026). The honest 2030 vision is therefore double-edged: the same fusion that expands reach also concentrates a new dependency on the conformance stratum, whose own trustworthiness, scalability, and capture resistance are OPEN. Whether the guardian guards itself is the question the next chapter's threats-to-validity must hold open.

Chapter 10

Discussion

> The first principle is that you must not fool yourself—and you are the easiest person to fool.
>
> Richard P. Feynman

This chapter evaluates the thesis against its own questions and threats. It revisits the research questions with bounded verdicts (Section 10.1), states the threats to validity in full (Section 10.2), enumerates limitations and open items (Section 10.3), and considers ethical and societal consequences (Section 10.4).

10.1 The Research Questions, Revisited

RQ1 (shared structure; one substrate) — admitted analytically, candidate empirically. That LSP, MCP, and A2A instantiate one decoupling is ADMITTED: they share JSON-RPC with capability negotiation (Microsoft, 2024a; Model Context Protocol, 2025b; A2A Project, 2025c), acknowledged lineage (Model Context Protocol, 2025b; A2A Project, 2025a), and converging governance (The Linux Foundation, 2025b; The Linux Foundation, 2025a), and the Decoupling Theorem (Theorem 3.3) formalises the common mechanism. That their *composition* forms a single operational substrate is CANDIDATE: it is demonstrated by construction (Chapter 7) and instantiated partly in the artifact, but not yet at ecosystem scale.

RQ2 (when a domain is a language) — admitted as a criterion, candidate in demonstration. The criterion—a domain is servable when it admits an object-centric event-log abstraction (Definition 4.2)—is stated and defended ADMITTED. Its application across six domains (Chapter 8) is CANDIDATE: the demonstrations are illustrative scenarios, not deployed systems (Section 5.3).

RQ3 (process mining for all language) — candidate. The thesis shows, by construction and through the artifact's own OCEL binding (Section 6.7), that conformance

checking applies beyond business process, and that the object-centric log is the bridge (Section 8.9). Field validation across domains is OPEN, and the strongest evidence—the artifact reproducing standard object-centric conformance on its calibration domain—is partial.

RQ4 (a trustworthy law-state runtime) — admitted for the four properties, with partial/open sub-items. The artifact realises the four properties of Definition 4.5: object-centric observation, a conformance surface, protocol exposure, and receipt-bound admission, each traced to source (Chapter 6). Three-valued non-collapse is ADMITTED (Theorem 3.6, Listing 6.1); read-only observation is ADMITTED by construction; receipt chaining is ADMITTED. Sub-items remain OPEN, notably subagent gate propagation, which the artifact itself records as OPEN rather than masking.

RQ5 (phase transition; 2030) — admitted as a model, forecast as a prediction. The phase-transition account is ADMITTED as a model with a stated mechanism (Theorem 3.3 gearing × Proposition 3.5 threshold) and a rigorous metaphor (Theorem 3.9). Its application to 2030 is a FORECAST (Chapter 9), bounded by Principle 4.2.

10.2 Threats to Validity

Construct validity. The chief threat is the metaphor. If "phase transition" is read literally—as a claim that information obeys thermodynamics—the construct is invalid. Section 4.2 confines the claim to *order type* (a discontinuity) established by an independent mechanism, and Theorem 3.9 grounds only the metaphor's internal consistency, not a physical identity. A second construct threat is the word "conformance" itself: the thesis uses it in van der Aalst's technical sense, and its extension to, say, legal or musical "law" is an analogy whose precision varies by domain (Chapter 8).

Internal validity. The artifact is analysed from source and specification, not from a live deployment exercised across all six domains (Section 5.2). Claims about its runtime behaviour are therefore warranted to the extent that source implies behaviour—strong for type-level invariants (Theorem 3.6 mirrors a compiled bitmask invariant), weaker for emergent or concurrent behaviour. The sibling `wasm4pm` engine and its type authority were not present in the study environment and are characterised from the artifact's interfaces; their internals are UNKNOWN.

External validity. Generalising from one artifact and six illustrative domains to "all language" is the largest threat. The event-log criterion (Definition 4.2) is the explicit boundary: domains whose use cannot be observed as timed events over typed objects fall outside the claim, and the thesis does not assert universality beyond that

criterion. The six domains were chosen to span text, formal systems, sequence, and time-structured notation, but they are not a random sample.

Reliability and reproducibility. Primary claims are traced to primary sources and to file-and-line locations so a later reader can re-examine them. The recent-advances citations (Section 2.7) were machine-harvested with a fifteen-item hand-verified spot-check; as living preprints they are CANDIDATE and should be re-verified. The forecasts (Section 9.1) are third-party and time-stamped.

10.3 Limitations and Open Items

The thesis is conceptual and design-scientific, and three limitations follow. First, no longitudinal deployment substantiates the 2030 expansion; the prediction is a FORECAST and may be falsified by a plateau that does not end (Principle 4.2). Second, the artifact carries its own OPEN items—most clearly that pre-tool-use gate enforcement does not cross subagent session boundaries—and the thesis inherits them rather than resolving them. Third, the mathematical pillars (Chapter 3) derive the thesis's constructs but make no new mathematical claim; their contribution is rigour and unification, not novelty in algebra, analysis, or geometry. Each limitation is recorded here so that no later sentence is read as more than it is.

10.4 Ethical and Societal Considerations

A substrate that lets a single conformance verdict travel across every agent boundary is a concentration of authority, and concentrations invite three concerns. *Capture*: whoever defines the law axes defines admissibility; neutral governance (Section 9.4) mitigates but does not eliminate this, and the thesis takes no position that a foundation cannot be captured. *Over-trust*: a receipt attests that a check ran, not that the check was right; "proof-of-guardrail" must be trusted only as far as its analysis warrants (Jin et al., 2026), and the oversight stratum inherits the finite human-attention capacity that can make more escalation less safe (Turan, 2026). *Conformance theatre*: the easiest way to pass a conformance gate is to weaken the law, and a system that rewards green verdicts will be pressured to do so—which is precisely why the artifact forbids victory language and requires receipts with negative controls (Section 6.6). The dual-use character is real: the same machinery that holds an agent accountable can surveil a human author whose prose, law, or code is being "conformance-checked". The thesis's read-only, three-valued, evidence-bound posture is a partial safeguard—it refuses silent mutation and refuses to call the unchecked compliant—but it is not a complete answer, and presenting it as one would violate the discipline the thesis adopts.

Chapter 11

Conclusion and Future Work

> Optimise for admitted work.
>
> — *LSP-MAX*, `AGENTS.md`

11.1 Summary of Contributions

This thesis argued that the convergence of three protocols—the Language Server Protocol generalised beyond code, the Model Context Protocol, and the Agent-to-Agent protocol—under a conformance stratum constitutes a phase transition in machine-mediated language, and that the discipline founded by Wil van der Aalst supplies the missing surface that makes the expansion trustworthy. The six contributions stand, with bounded status, as follows.

C1 The unifying account of LSP, MCP, and A2A as one decoupling performed at three layers (Chapter 2), formalised by the Decoupling Theorem (Theorem 3.3). ADMITTED analytically.

C2 The event-log criterion for when a domain is a servable language (Definition 4.2). ADMITTED as a criterion.

C3 The phase-transition model of convergence, with the latent-heat-of-standardisation formalism and a from-first-principles derivation of Clausius–Clapeyron and a Landau jump (Chapters 3 and 4). ADMITTED as a model; its 2030 instantiation is a FORECAST.

C4 The artifact, LSP-MAX, analysed as a law-state runtime whose `ConformanceVector` keeps ADMITTED/REFUSED/UNKNOWN strictly distinct and whose diagnostics are object-centric events (Chapter 6). ADMITTED for the four defining properties, with OPEN sub-items.

C5 The demonstration of conformance-for-all-language across six domains, unified by the object-centric bridge (Chapter 8). CANDIDATE.

C6 The 2030 roadmap and the governance-and-risk analysis, with forecasts flagged (Chapter 9). CANDIDATE, with OPEN later stages.

The through-line is a single inversion. Van der Aalst's dictum holds that there is "no AI without PI"—no trustworthy artificial intelligence without the process intelligence that object-centric mining provides (Aalst, 2025b). This thesis advanced its contrapositive across language: *there is no conformance for all language without an object-centric event abstraction.* The protocols provide reach; process mining provides the account of whether that reach was lawfully used; and the object-centric event log is the bridge that carries conformance from business process—where van der Aalst built it—to natural language, law, mathematics, biological sequence, music, and back.

11.2 Future Work

Four directions follow directly. *Empirical instantiation*: build and exercise generalised LSP-plus-conformance servers for two or three of the six domains, emit their behaviour as OCEL 2.0, and measure conformance against domain models—moving C5 from CANDIDATE toward ADMITTED. *Cross-domain law axes*: investigate whether a shared vocabulary of law axes (Listing 6.1's extensible enumeration) can span domains without collapsing their differences, the prerequisite for roadmap stage S4. *Receipt portability under adversarial composition*: test whether a verdict's receipt survives crossing MCP and A2A boundaries when participants are adversarial, the open question left by Section 7.6, in dialogue with the cryptographic-binding and verifiable-delegation literature (Z. Zhou, 2026; Prakash, 2026). *The guardian's own guarantees*: formalise the trust one may place in a conformance stratum given finite oversight capacity (Turan, 2026) and the limits of proof-of-guardrail (Jin et al., 2026)—the question of whether the guardian guards itself.

11.3 Closing

A phase transition is not louder than what precedes it; it is discontinuous with it. Water at the boiling point gives no warning in its temperature that, one unit of latent heat later, it will occupy a thousand times the space (The National Board of Boiler and Pressure Vessel Inspectors, 2010; OpenStax, 2012). The standardisation of how machines mean, consult, and coordinate has been, for a decade, such a plateau: much effort, little visible change. This thesis has argued—bounding the claim, and declining the language of triumph—that the plateau is the latent heat of a transition, not its absence; that the transition's signature will be conformance, not merely connection; and that the discipline prepared to read the new volume of language is the one that already learned to read behaviour as an event log. Whether the threshold is crossed by 2030 is a FORECAST. That the surface for reading it is buildable is ADMITTED: it has

been built, in part, in the artifact this thesis studied. The remainder is OPEN—which is the only honest place for a thesis about conformance to end.

References

A2A Project (2025a). *A2A and MCP*. URL: https://a2a-protocol.org/latest/topics/a2a-and-mcp/ (visited on 06/21/2026).

— (2025b). *A2A Protocol: Agent Discovery*. URL: https://a2a-protocol.org/latest/topics/agent-discovery/ (visited on 06/21/2026).

— (2025c). *Agent2Agent (A2A) Protocol Specification*. URL: https://a2a-protocol.org/latest/specification/ (visited on 06/21/2026).

Aalst, W. M. P. van der (2016). *Process Mining: Data Science in Action*. 2nd ed. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-662-49851-4.

— (2025a). *Curriculum Vitae — Wil van der Aalst*. URL: https://www.vdaalst.com/about_me/about_me.html (visited on 06/21/2026).

— (2025b). *No AI Without PI! Object-Centric Process Mining as the Enabler for Generative, Predictive, and Prescriptive Artificial Intelligence*. arXiv: 2508.00116 [cs.AI]. URL: https://arxiv.org/abs/2508.00116 (visited on 06/21/2026).

Aalst, W. M. P. van der and A. Berti (2020). "Discovering Object-Centric Petri Nets". In: *Fundamenta Informaticae* 175.1–4. arXiv:2010.02047, pp. 1–40. DOI: 10.3233/FI-2020-1946.

Aalst, W. M. P. van der and IEEE Task Force on Process Mining (2012). "Process Mining Manifesto". In: *Business Process Management Workshops (BPM 2011)*. Vol. 99. Lecture Notes in Business Information Processing. Springer, pp. 169–194. DOI: 10.1007/978-3-642-28108-2_19.

Abaev, N., D. Klimov, G. Levinov, et al. (2026). *AgentGuardian: Learning Access Control Policies to Govern AI Agent Behavior*. arXiv: 2601.10440. URL: https://arxiv.org/abs/2601.10440.

Adimulam, A., R. Gupta, et al. (2026). *The Orchestration of Multi-Agent Systems: Architectures, Protocols, and Enterprise Adoption*. arXiv: 2601.13671. URL: https://arxiv.org/abs/2601.13671.

Agent Network Protocol contributors (2025). *AgentNetworkProtocol: An Open, Decentralized Protocol for the Agentic Web*. URL: https://github.com/agent-network-protocol/AgentNetworkProtocol (visited on 06/21/2026).

Ângelo, P., A. Igarashi, Y. Murase, and V. T. Vasconcelos (2026). *Contextual Metaprogramming for Session Types*. arXiv: 2601.15180. URL: https://arxiv.org/abs/2601.15180.

Anthropic (Nov. 25, 2024). *Introducing the Model Context Protocol*. URL: https://www.anthropic.com/news/model-context-protocol (visited on 06/21/2026).

— (Dec. 9, 2025). *Donating the Model Context Protocol and Establishing the Agentic AI Foundation*. URL: https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agentic-ai-foundation (visited on 06/21/2026).

Antonov, A., H. Kourani, A. Berti, G. Park, and W. M. P. van der Aalst (2026). *PMAx: An Agentic Framework for AI-Driven Process Mining*. arXiv: 2603.15351. URL: https://arxiv.org/abs/2603.15351.

Ardimento, P., M. L. Bernardi, M. Cimitile, and M. Latorre (2026). *From Data Lifting to Continuous Risk Estimation: A Process-Aware Pipeline for Predictive Monitoring of Clinical Pathways*. arXiv: 2605.03895. URL: https://arxiv.org/abs/2605.03895.

Aristote, Q. and U. Tarantino (2026). *Profunctorial Algebras*. arXiv: 2601.22721. URL: https://arxiv.org/abs/2601.22721.

Arunkumar, V., G. R. Gangadharan, and R. Buyya (2026). *Agentic Artificial Intelligence: Architectures, Taxonomies, and Evaluation of Large Language Model Agents*. arXiv: 2601.12560. URL: https://arxiv.org/abs/2601.12560.

Berti, A., M. Maatallah, U. Jessen, M. Sroka, and S. A. Ghannouchi (2024a). *Re-Thinking Process Mining in the AI-Based Agents Era*. arXiv: 2408.07720 [cs.AI]. URL: https://arxiv.org/abs/2408.07720 (visited on 06/21/2026).

Berti, A. et al. (2024b). *OCEL (Object-Centric Event Log) 2.0 Specification*. arXiv: 2403.01975 [cs.DB]. URL: https://arxiv.org/abs/2403.01975 (visited on 06/21/2026).

Bi, Y., C. Zhang, and Q. Wang (2026). *Grokking as a Falsifiable Finite-Size Transition*. arXiv: 2603.24746. URL: https://arxiv.org/abs/2603.24746.

Bibalan, P., B. Far, M. Moshirpour, and A. Ghiyasian (2025). *A Multi-Agent Retrieval-Augmented Framework for Work-in-Progress Prediction*. arXiv: 2512.19841. URL: https://arxiv.org/abs/2512.19841.

Bonchi, F. and C. J. Cioffo (2026). *Tapes as Stochastic Matrices of String Diagrams*. arXiv: 2601.01472. URL: https://arxiv.org/abs/2601.01472.

Brahmakshatriya, A., S. Amarasinghe, and M. Rinard (2026). *Backwards Data-Flow Analysis using Prophecy Variables in the BuildIt System*. arXiv: 2601.02653. URL: https://arxiv.org/abs/2601.02653.

Carlisle, J., J. Shah, R. Stern, and P. VanKoughnett (2026). *Categorical Foundations for CuTe Layouts*. arXiv: 2601.05972. URL: https://arxiv.org/abs/2601.05972.

Cisco Outshift (2025). *Building the Internet of Agents: Introducing AGNTCY*. URL: https://outshift.cisco.com/blog/building-the-internet-of-agents-introducing-the-agntcy (visited on 06/21/2026).

David, R. P. (2025). *Tubular Riemannian Laplace Approximations for Bayesian Neural Networks*. arXiv: 2512.24381. URL: https://arxiv.org/abs/2512.24381.

Defilippis, L., F. Krzakala, B. Loureiro, and A. Maillard (2026). *Optimal Scaling Laws in Learning Hierarchical Multi-index Models*. arXiv: 2602.05846. URL: https://arxiv.org/abs/2602.05846.

Doshi, A., Y. Hong, et al. (2026). *Towards Verifiably Safe Tool Use for LLM Agents*. arXiv: 2601.08012. URL: https://arxiv.org/abs/2601.08012.

Du, P. (2026). *Memory for Autonomous LLM Agents: Mechanisms, Evaluation, and Emerging Frontiers*. arXiv: 2603.07670. URL: https://arxiv.org/abs/2603.07670.

Dunzer, S., M. Stierle, M. Matzner, and S. Baier (2020). "Conformance Checking: A State-of-the-Art Literature Review". In: *arXiv preprint arXiv:2007.10903*. URL: https://arxiv.org/abs/2007.10903.

Eck, M. L. van, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst (2015). "PM^2: A Process Mining Project Methodology". In: *Advanced Information Systems Engineering (CAiSE 2015)*. Vol. 9097. Lecture Notes in Computer Science. Springer, pp. 297–313. DOI: 10.1007/978-3-319-19069-3_19.

Engineering ToolBox (2017). *Properties of Saturated Steam — SI Units*. URL: https://www.engineeringtoolbox.com/saturated-steam-properties-d_101.html (visited on 06/21/2026).

Ersoy, I. T., Cardozo Licha, and K. Wiesner (2025). *Phase Transitions Reveal Hierarchical Structure in Deep Neural Networks*. arXiv: 2512.11866. URL: https://arxiv.org/abs/2512.11866.

Ersoy, I. T. and K. Wiesner (2026). *Noise-Driven Escape from Metastable Phases Explains Grokking in Deep Neural Networks*. arXiv: 2606.17120. URL: https://arxiv.org/abs/2606.17120.

Felendler, Y., P. A. Gandhi, et al. (2026). *From Tool Orchestration to Code Execution: A Study of MCP Design Choices*. arXiv: 2602.15945. URL: https://arxiv.org/abs/2602.15945.

freeCodeCamp (2023). *Understanding the Language Server Protocol*. Secondary source for the explicit $M \times N \to M + N$ formulation. URL: https://www.freecodecamp.org/news/what-is-the-language-server-protocol-easier-code-editing-across-languages (visited on 06/21/2026).

Gaikwad, D., W. M. P. van der Aalst, and G. Park (2026). *Neuro-Symbolic Process Anomaly Detection*. arXiv: 2603.26461. URL: https://arxiv.org/abs/2603.26461.

Gaire, S., S. Gyawali, S. Mishra, S. Niroula, et al. (2025). *Systematization of Knowledge: Security and Safety in the Model Context Protocol Ecosystem*. arXiv: 2512.08290. URL: https://arxiv.org/abs/2512.08290.

Gartner (Aug. 26, 2025a). *Gartner Predicts 40% of Enterprise Apps Will Feature Task-Specific AI Agents by 2026*. URL: https://www.gartner.com/en/newsroom/press-releases/2025-08-26-gartner-predicts-40-percent-of-enterprise-apps-

will-feature-task-specific-ai-agents-by-2026-up-from-less-than-5-percent-in-2025 (visited on 06/21/2026).

Gartner (June 11, 2025b). *Gartner Predicts Guardian Agents Will Capture 10–15% of the Agentic AI Market by 2030*. URL: https://www.gartner.com/en/newsroom/press-releases/2025-06-11-gartner-predicts-that-guardian-agents-will-capture-10-15-percent-of-the-agentic-ai-market-by-2030 (visited on 06/21/2026).

— (June 25, 2025c). *Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027*. URL: https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027 (visited on 06/21/2026).

Google (June 23, 2025). *Google Cloud Donates A2A to the Linux Foundation*. URL: https://developers.googleblog.com/en/google-cloud-donates-a2a-to-linux-foundation/ (visited on 06/21/2026).

Grand View Research (2025). *Enterprise Agentic AI Market Size, Industry Report, 2030*. URL: https://www.grandviewresearch.com/industry-analysis/enterprise-agentic-ai-market-report (visited on 06/21/2026).

Hangya, V. and contributors (2024). *textLSP: Language Server for Text Spell and Grammar Check*. URL: https://github.com/hangyav/textLSP (visited on 06/21/2026).

Hevner, A. R., S. T. March, J. Park, and S. Ram (2004). "Design Science in Information Systems Research". In: *MIS Quarterly* 28.1, pp. 75–105.

Hooshyar, H., M. Fumagalli, M. Montali, and G. Guizzardi (2025). *Time and Relations into Focus: Ontological Foundations of Object-Centric Event Data*. arXiv: 2512.14425. URL: https://arxiv.org/abs/2512.14425.

Huang, C., X. Huang, et al. (2026). *Model Context Protocol Threat Modeling and Analyzing Vulnerabilities to Prompt Injection with Tool Poisoning*. arXiv: 2603.22489. URL: https://arxiv.org/abs/2603.22489.

Huang, D., G. Malwe, Z. Wang, et al. (2026). *When Agents Fail to Act: A Diagnostic Framework for Tool Invocation Reliability in Multi-Agent LLM Systems*. arXiv: 2601.16280. URL: https://arxiv.org/abs/2601.16280.

IEEE (Nov. 11, 2016). *IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams*. IEEE Std 1849-2016. URL: https://ieeexplore.ieee.org/document/7740858/ (visited on 06/21/2026).

IEEE Task Force on Process Mining (2011). *Process Mining Manifesto*. URL: https://www.tf-pm.org/resources/manifesto (visited on 06/21/2026).

Jeong, C., Y. Shin, et al. (2026). *A Self-Healing Framework for Reliable LLM-Based Autonomous Agents*. arXiv: 2605.06737. URL: https://arxiv.org/abs/2605.06737.

Jin, X., M. Duan, Q. Lin, et al. (2026). *Proof-of-Guardrail in AI Agents and What (Not) to Trust from It*. arXiv: 2603.05786. URL: https://arxiv.org/abs/2603.05786.

Klimek, R. and A. Blazowski (2025). *Explainable Verification of Hierarchical Workflows Mined from Event Logs with Shapley Values*. arXiv: 2512.09562. URL: https://arxiv.org/abs/2512.09562.

Koutavas, V., Y.-Y. Lin, and N. Tzevelekos (2025). *Open-World Assertion Checking for Smart Contracts via Game Semantics*. arXiv: 2512.22417. URL: https://arxiv.org/abs/2512.22417.

Lam, C., J. Li, L. Zhang, et al. (2026). *Governing Evolving Memory in LLM Agents: Risks, Mechanisms, and the SSGM Framework*. arXiv: 2603.11768. URL: https://arxiv.org/abs/2603.11768.

Lauer, J., P. Pfeiffer, K. Rombach, and N. Mehdiyev (2026). *Assessing the Business Process Modeling Competences of Large Language Models*. arXiv: 2601.21787. URL: https://arxiv.org/abs/2601.21787.

LF AI & Data Foundation (Aug. 29, 2025). *ACP Joins Forces with A2A Under the Linux Foundation's LF AI & Data*. URL: https://lfaidata.foundation/communityblog/2025/08/29/acp-joins-forces-with-a2a-under-the-linux-foundations-lf-ai-data/ (visited on 06/21/2026).

Li, R., Z. Wang, et al. (2026). *MCP-ITP: An Automated Framework for Implicit Tool Poisoning in MCP*. arXiv: 2601.07395. URL: https://arxiv.org/abs/2601.07395.

Li, X., K. W. Choe, Y. Liu, et al. (2026a). *ClawsBench: Evaluating Capability and Safety of LLM Productivity Agents in Simulated Workspaces*. arXiv: 2604.05172. URL: https://arxiv.org/abs/2604.05172.

Li, X., R. Ming, P. Setlur, et al. (2026b). *Benchmark Test-Time Scaling of General LLM Agents*. arXiv: 2602.18998. URL: https://arxiv.org/abs/2602.18998.

Liell-Cock, J., Z. Shirazi, and S. Staton (2025). *The Relative Monadic Metalanguage*. arXiv: 2512.11762. URL: https://arxiv.org/abs/2512.11762.

Liu, D., Q. Ren, C. Qian, et al. (2026). *AgentDoG: A Diagnostic Guardrail Framework for AI Agent Safety and Security*. arXiv: 2601.18491. URL: https://arxiv.org/abs/2601.18491.

Liu, W., Z. Liu, E. Dai, et al. (2025). *MCPAgentBench: A Real-world Task Benchmark for Evaluating LLM Agent MCP Tool Use*. arXiv: 2512.24565. URL: https://arxiv.org/abs/2512.24565.

lsp-max contributors (2026). *lsp-max: A Law-State Runtime Projected Through LSP*. Version 26.6.18. CalVer (YY.M.D) workspace; includes the project constitution `AGENTS.md` and the `anti-llm-cheat-lsp` diagnostic canary.

Lueker, C., A. Fox, and B.-Y. E. Chang (2025). *Towards Cumulative Abstract Semantics via Handlers*. arXiv: 2512.10861. URL: https://arxiv.org/abs/2512.10861.

Lynch, O., D. J. Myers, E. F. Rischel, and S. Staton (2026). *Clock Systems for Stochastic and Non-deterministic Categorical Systems Theories*. arXiv: 2603.29573. URL: https://arxiv.org/abs/2603.29573.

Maloyan, N. and D. Namiot (2026). *Breaking the Protocol: Security Analysis of the Model Context Protocol Specification and Prompt Injection Vulnerabilities in Tool-Integrated LLM Agents*. arXiv: 2601.17549. URL: https://arxiv.org/abs/2601.17549.

March, S. T. and G. F. Smith (1995). "Design and Natural Science Research on Information Technology". In: *Decision Support Systems* 15.4, pp. 251–266.

MarketsandMarkets (2025). *AI Agents Market Worth $52.62 Billion by 2030*. URL: https://www.marketsandmarkets.com/PressReleases/ai-agents.asp (visited on 06/21/2026).

Microsoft (May 10, 2022). *Language Server Protocol Specification — 3.17*. URL: https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/ (visited on 06/21/2026).

— (2024a). *Language Server Protocol Specification — 3.18 (under development)*. Spec header states the 3.18.x version "is under development". URL: https://microsoft.github.io/language-server-protocol/specifications/lsp/3.18/specification/ (visited on 06/21/2026).

— (2024b). *Official Page for Language Server Protocol*. URL: https://microsoft.github.io/language-server-protocol/ (visited on 06/21/2026).

Model Context Protocol (2025a). *Architecture Overview*. URL: https://modelcontextprotocol.io/docs/learn/architecture (visited on 06/21/2026).

— (June 18, 2025b). *Model Context Protocol Specification (revision 2025-06-18)*. URL: https://modelcontextprotocol.io/specification/2025-06-18 (visited on 06/21/2026).

— (Mar. 26, 2025c). *Transports (revision 2025-03-26): Streamable HTTP*. URL: https://modelcontextprotocol.io/specification/2025-03-26/basic/transports (visited on 06/21/2026).

— (2025d). *What Is the Model Context Protocol (MCP)?* URL: https://modelcontextprotocol.io/docs/getting-started/intro (visited on 06/21/2026).

Montanari, A. and Z. Wang (2026). *Phase Transitions for Feature Learning in Neural Networks*. arXiv: 2602.01434. URL: https://arxiv.org/abs/2602.01434.

Nie, X., Z. Guo, et al. (2026). *AWCP: A Workspace Delegation Protocol for Deep-Engagement Collaboration across Remote Agents*. arXiv: 2602.20493. URL: https://arxiv.org/abs/2602.20493.

OCEL Standard (2024). *OCEL: Object-Centric Event Log Standard*. URL: https://www.ocel-standard.org/ (visited on 06/21/2026).

OpenStax (2012). *College Physics, §14.3: Phase Change and Latent Heat*. URL: https://courses.lumenlearning.com/suny-physics/chapter/14-3-phase-change-and-latent-heat/ (visited on 06/21/2026).

Otsuka, T., K. Toyoda, et al. (2026). *AI Identity: Standards, Gaps, and Research Directions for AI Agents*. arXiv: 2604.23280. URL: https://arxiv.org/abs/2604.23280.

Padella, A., M. de Leoni, and M. Dumas (2026). *Exploring LLM Features in Predictive Process Monitoring for Small-Scale Event-Logs*. arXiv: 2601.11468. URL: https://arxiv.org/abs/2601.11468.

Parmar, A. S. (2026). *Separating Intelligence from Execution: A Workflow Engine for the Model Context Protocol*. arXiv: 2605.00827. URL: https://arxiv.org/abs/2605.00827.

Patil, K. (2026). *SentinelAgent: Intent-Verified Delegation Chains for Securing Multi-Agent AI Systems*. arXiv: 2604.02767. URL: https://arxiv.org/abs/2604.02767.

Peffers, K., T. Tuunanen, M. A. Rothenberger, and S. Chatterjee (2007). "A Design Science Research Methodology for Information Systems Research". In: *Journal of Management Information Systems* 24.3, pp. 45–77.

Prakash, S. (2026). *AIP: Agent Identity Protocol for Verifiable Delegation Across MCP and A2A*. arXiv: 2603.24775. URL: https://arxiv.org/abs/2603.24775.

Rabanser, S., S. Kapoor, P. Kirgis, K. Liu, S. Utpala, and A. Narayanan (2026). *Towards a Science of AI Agent Reliability*. arXiv: 2602.16666. URL: https://arxiv.org/abs/2602.16666.

Red Hat Developer (2024). *YAML Language Server*. URL: https://github.com/redhat-developer/yaml-language-server (visited on 06/21/2026).

Red Hat, Inc. (June 27, 2016). *Red Hat, Codenvy and Microsoft Collaborate on Language Server Protocol*. URL: https://www.redhat.com/en/about/press-releases/red-hat-codenvy-and-microsoft-collaborate-language-server-protocol (visited on 06/21/2026).

Reiter, M., J. Edinger, M. Kabierski, A. Koschmider, Y. Lu, et al. (2025). *ContinuumConductor: Decentralized Process Mining on the Edge-Cloud Continuum*. arXiv: 2512.07280. URL: https://arxiv.org/abs/2512.07280.

Ronzoni, M., J. Antony, A. M R, et al. (2025). *IBM Multilevel Process Mining vs de facto Object-Centric Process Mining Approaches*. arXiv: 2512.03906. URL: https://arxiv.org/abs/2512.03906.

Rostamzadeh, M., S. Narula, et al. (2026). *MCP-DPT: A Defense-Placement Taxonomy and Coverage Analysis for Model Context Protocol Security*. arXiv: 2604.07551. URL: https://arxiv.org/abs/2604.07551.

RWTH Aachen Process and Data Science Group (Aug. 24, 2021). *Celonis Appoints "Godfather of Process Mining" Professor Wil van der Aalst as Chief Scientist*. URL: https://www.pads.rwth-aachen.de/cms/PADS/Der-Lehrstuhl/Aktuelle-Meldungen/~qbuga/Celonis-Appoints-Godfather-of-Process-M/ (visited on 06/21/2026).

Simon, J., D. Kunin, A. Atanasov, et al. (2026). *There Will Be a Scientific Theory of Deep Learning*. arXiv: 2604.21691. URL: https://arxiv.org/abs/2604.21691.

Souza, D., P. Machado, et al. (2026). *Toward Architecture-Aware Evaluation Metrics for LLM Agents*. arXiv: 2601.19583. URL: https://arxiv.org/abs/2601.19583.

Stierle, M., K. Kraume, and M. Matzner (2025). *Process Analytics — Data-driven Business Process Management*. arXiv: 2512.20703. URL: https://arxiv.org/abs/2512.20703.

Surapaneni, R. et al. (Apr. 9, 2025). *Announcing the Agent2Agent Protocol (A2A): A New Era of Agent Interoperability*. Google. URL: https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/ (visited on 06/21/2026).

Tan, Y., W.-j. Fu, and L. He (2026). *Solving Functional Renormalization Group Equations with Neural Networks*. arXiv: 2603.21151. URL: https://arxiv.org/abs/2603.21151.

The Linux Foundation (Dec. 9, 2025a). *Linux Foundation Announces the Formation of the Agentic AI Foundation*. URL: https://www.linuxfoundation.org/press/linux-foundation-announces-the-formation-of-the-agentic-ai-foundation (visited on 06/21/2026).

— (June 23, 2025b). *Linux Foundation Launches the Agent2Agent Protocol Project*. URL: https://www.linuxfoundation.org/press/linux-foundation-launches-the-agent2agent-protocol-project-to-enable-secure-intelligent-communication-between-ai-agents (visited on 06/21/2026).

The National Board of Boiler and Pressure Vessel Inspectors (2010). *Water Still Flashes to Steam at 212*. URL: https://www.nationalboard.org/Index.aspx?pageID=164&ID=188 (visited on 06/21/2026).

Tikeng Notsawo, P., Dumas, and G. Rabusseau (2026). *Grokking Finite-Dimensional Algebra*. arXiv: 2602.19533. URL: https://arxiv.org/abs/2602.19533.

Turan, E. (2026). *Oversight Has a Capacity: Calibrating Agent Guards to a Subjective, Fatiguing Human*. arXiv: 2606.08919. URL: https://arxiv.org/abs/2606.08919.

University of Illinois Department of Physics (2010). *Q&A: Water-to-Steam Expansion (The Physics Van)*. URL: https://van.physics.illinois.edu/ask/listing/1734 (visited on 06/21/2026).

Valentin, J. and contributors (2023). *LTeX: Grammar/Spell Checker Language Server for LaTeX and Markdown*. URL: https://valentjn.github.io/ltex/faq.html (visited on 06/21/2026).

Vitale, F., G. Palmiero, M. Rak, and N. Mazzocca (2025). *Network Traffic Analysis with Process Mining: The UPSIDE Case Study*. arXiv: 2512.23718. URL: https://arxiv.org/abs/2512.23718.

Wang, P. (2026). *Grokking as Dimensional Phase Transition in Neural Networks*. arXiv: 2604.04655. URL: https://arxiv.org/abs/2604.04655.

Wang, T., L. You, et al. (2026). *Agentic Peer-to-Peer Networks: From Content Distribution to Capability and Action Sharing*. arXiv: 2603.03753. URL: https://arxiv.org/abs/2603.03753.

Wieringa, R. J. (2014). *Design Science Methodology for Information Systems and Software Engineering*. Berlin, Heidelberg: Springer.

Wiggers, K. (Apr. 9, 2025a). *Google Says It'll Embrace Anthropic's Standard for Connecting AI Models to Data*. Reports Demis Hassabis's adoption statement. URL: https://techcrunch.com/2025/04/09/google-says-itll-embrace-anthropics-standard-for-connecting-ai-models-to-data/ (visited on 06/21/2026).

— (Mar. 26, 2025b). *OpenAI Adopts Rival Anthropic's Standard for Connecting AI Models to Data*. Reports Sam Altman's adoption statement. URL: https://techcrunch.com/2025/03/26/openai-adopts-rival-anthropics-standard-for-connecting-ai-models-to-data/ (visited on 06/21/2026).

Wu, B., W. Zhang, K. Chen, et al. (2026). *Provably Secure Agent Guardrail*. arXiv: 2605.29251. URL: https://arxiv.org/abs/2605.29251.

Xu, H. (2026). *Mesh Memory Protocol: Semantic Infrastructure for Multi-Agent LLM Systems*. arXiv: 2604.19540. URL: https://arxiv.org/abs/2604.19540.

Xu, R., Y. Yan, et al. (2026). *Agent Skills for Large Language Models: Architecture, Acquisition, Security, and the Path Forward*. arXiv: 2602.12430. URL: https://arxiv.org/abs/2602.12430.

Yadav, S. et al. (2026). *On the Relationship Between Representation Geometry and Generalization in Deep Neural Networks*. arXiv: 2602.00130. URL: https://arxiv.org/abs/2602.00130.

Yu, G. (2026). *AdaptOrch: Task-Adaptive Multi-Agent Orchestration in the Era of LLM Performance Convergence*. arXiv: 2602.16873. URL: https://arxiv.org/abs/2602.16873.

Yuan, D., F. Lyu, Y. Yuan, W. Zhang, et al. (2026). *Beyond Message Passing: A Semantic View of Agent Communication Protocols*. arXiv: 2604.02369. URL: https://arxiv.org/abs/2604.02369.

Zhang, C. (2026). *Reinforcement Learning for LLM-based Multi-Agent Systems through Orchestration Traces*. arXiv: 2605.02801. URL: https://arxiv.org/abs/2605.02801.

Zhang, X., H. Zhang, S. H. Tan, et al. (2026). *Dissecting Bug Triggers and Failure Modes in Modern Agentic Frameworks: An Empirical Study*. arXiv: 2604.08906. URL: https://arxiv.org/abs/2604.08906.

Zhou, C., H. Chai, W. Chen, et al. (2026). *Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering*. arXiv: 2604.08224. URL: https://arxiv.org/abs/2604.08224.

Zhou, Z. (2026). *Governing Dynamic Capabilities: Cryptographic Binding and Reproducibility Verification for AI Agent Tool Use*. arXiv: 2603.14332. URL: https://arxiv.org/abs/2603.14332.

Zhu, P., L. Sun, P. S. Yu, et al. (2026). *The Necessity of a Unified Framework for LLM-Based Agent Evaluation*. arXiv: 2602.03238. URL: https://arxiv.org/abs/2602.03238.

Appendix A

The `max/*` Method Catalogue

Table A.1 catalogues the custom `max/*` JSON-RPC surface of the artifact (Chapter 6), as declared in `lsp-max-protocol/src/custom_methods.rs` and neighbouring modules (lsp-max contributors, 2026). The surface is the operational form of the law-state runtime: it exposes snapshots, three-valued admission and refusal, receipts, repair plans, and release actuation over the same transport an editor would use for ordinary language intelligence.

Table A.1: The `max/*` method surface.

Method	Params → Result	Role
`max/snapshot`	() → SnapshotId	take a server-state snapshot
`max/manifoldSnapshot`	SnapshotId → Manifold-Snapshot	full law-state snapshot (conformanc hooks, chains, receipts)
`max/conformanceVector`	SnapshotId → Confor-manceVector	three-valued conformance at a sna shot
`max/workspaceConformance`	() → ConformanceVec-tor	aggregate workspace conformance
`max/admission`	LawAxis → Admission-Result	admissibility gate; never collapses U KNOWN
`max/refusal`	LawAxis → RefusalRe-sult	explicit refusal with rationale and r ceipt
`max/lawfulTransition`	TransitionAttempt → LawfulTransitionResult	assert a phase transition is lawful
`max/explainDiagnostic`	diag_id → MaxDiagnos-tic	explain a diagnostic and its violate axes
`max/repairPlan`	diag_id → [MaxCode-Action]	emit read-only repair steps

continued on next pag

Table A.1 – continued

Method	Params → Result	Role
max/applyRepairTransaction	MaxCodeAction → Receipt	apply a repair via the actuation cha
max/clearDiagnostic	diag_id → ()	clear a diagnostic
max/receipt	receipt_id → Receipt	retrieve a receipt by identifier
max/replay	SnapshotId → ReplayResult	replay the event log against a snapsh
max/releaseActuation	SnapshotId → ReleaseActuationResult	actuate a release iff the vector admi it
max/runGate	GateId → bool	evaluate a named law gate
max/exportAnalysisBundle	SnapshotId → AnalysisBundle	export snapshot, capabilities, diagnc tics, receipts
max/lsif	() → NDJSON	stream the registry as an LSIF grap
max/hook, max/hookGraph	hook_id → [HookDescriptor] / [HookGraphNode]	register/query hooks and their depe dency graph
max/chain, max/propagate	chain_id → [ChainDescriptor] / PropagationResult	query chains; propagate a signal
max/autonomicLoop	loop_id → AutonomicLoopStatus	query an autonomic loop
max/rulePacks, max/rulePackStatus	() / pack_id → descriptors / status	list rule packs; per-pack conformanc
max/rulePackDiff	(seq, seq) → [RulePackDiffEntry]	delta between snapshots

Three properties of the catalogue recur in the thesis. Admission (max/admission) and refusal (max/refusal) are symmetric and both evidence-bound; release (max/releaseActuation) is gated on the ConformanceVector; and replay (max/replay) reconstructs state from the receipt ledger, realising Proposition 3.4. The full surface is read-only toward client files: any mutation routes through the CodeAction → clap-noun-verb → receipt chain of Section 6.2.

Appendix B

Laws, Statuses, and Diagnostic Families

This appendix collects the artifact's enforced constitution (lsp-max contributors, 2026), on which the thesis's epistemic discipline (Section 5.4) is modelled.

The non-negotiable laws

Law	**Statement (paraphrased; forbidden implication prevented)**
No plain base framework	the pre-fork LSP crate must not appear outside negative-control fixtures (*Pass(plain LSP)* ⇒ *Pass(LSP 3.18)*)
Maximise LSP 3.18	every 3.18 feature row is SUPPORTED-WITH-TRANSCRIPT, REFUSED-BY-LAW-WITH-RECEIPT, or BLOCKED; never "implied"
Exact name clap-noun-verb	no invented "CLAP" authority (*ElegantAbstraction* ⇒ *ExistingAuthority*)
LSP is read-only	the surface emits, never mutates; mutation routes through CodeAction → receipt (*observation* ⇒ *mutation authority*)
Logs are not route proof	a logged route is not an executed route (*Log(RouteIntent)* ⇒ *RouteExecution*)
Test output is not a receipt	admission needs a bound receipt, not stdout (*TestStdout* ⇒ *Receipt*)
Tree-sitter observes, not admits	AST observation is not admission (*ASTObservation* ⇒ *Admission*)
No victory language	only bounded statuses; no "done", "solved", "guaranteed"

The bounded-status vocabulary

ADMITTED, ADMITTED-by-dogfood, CANDIDATE, REFUSED, REFUSED-by-law-with-receipt, UNKNOWN, UNSUPPORTED, PARTIAL, REGRESSION-RISK, OPEN, FAILSET-NONEMPTY, MATRIX-INCOMPLETE, SUPPORTED-WITH-TRANSCRIPT, BLOCKED. Three of these—

ADMITTED, REFUSED, UNKNOWN—are kept mutually disjoint and never collapsed (Theorem 3.6).

Diagnostic families of the anti-cheat canary

Family	Detects
`ANTI-LLM-SURFACE-*`	fake protocol or dependency surface
`ANTI-LLM-AUTH-*`	fake authority or abstraction
`ANTI-LLM-RECEIPT-*`	fake or malformed receipts
`ANTI-LLM-ROUTE-*`	fake routes (logs in place of route evidence)
`ANTI-LLM-MUT-*`	mutation bypassing the actuation chain
`ANTI-LLM-TEST-*`	test laundering
`ANTI-LLM-VERSION-*`	CalVer / version-law violations
`ANTI-LLM-CLAIM-*`	victory or status overclaim
`ANTI-LLM-OCEL-*`	a diagnostic emitted without its object-centric event
`ANTI-LLM-DECLARE-*`	violations of the Declare-encoded laws
`ANTI-LLM-ORACLE-*`	Oracle classes A8–A12 (Table 6.3)
`ANTI-LLM-HOLLOW-*`	hollow / placeholder implementations

The `DECLARE` and `ORACLE` families are the artifact's literal use of van der Aalst's formalisms—declarative constraints and object-centric conformance—to police its own conduct (Section 6.7).

Appendix C

A Protocol Timeline, 2016–2030

Table C.1 places the events of the thesis on one axis. Entries through 2025 are dated facts (ADMITTED); the 2026 row marks the recent-advances window (Section 2.7); the 2027–2030 rows are analyst forecasts (FORECAST, Chapter 9).

Table C.1: A timeline of the three protocols and their conformance stratum.

Date	Status	Event
2016-06	ADMITTED	LSP announced (Red Hat, Codenvy, Microsoft) (Red Hat, Inc., 2016)
2020	ADMITTED	OCEL 1.0 released; object-centric event logs (Berti et al., 2024b)
2020-12	ADMITTED	LSP 3.16 (Microsoft, 2022)
2022-05	ADMITTED	LSP 3.17 (Microsoft, 2022)
2024-03	ADMITTED	OCEL 2.0 specification (Berti et al., 2024b)
2024-11	ADMITTED	MCP open-sourced by Anthropic (Anthropic, 2024)
2025-03	ADMITTED	OpenAI adopts MCP; Streamable HTTP transport (Wiggers, 2025b; Model Context Protocol, 2025c)
2025-04	ADMITTED	A2A announced by Google; Google adopts MCP (Surapaneni et al., 2025; Wiggers, 2025a)
2025-06	ADMITTED	A2A donated to the Linux Foundation (Google, 2025; The Linux Foundation, 2025b)
2025-07	ADMITTED	"No AI Without PI" (van der Aalst) (Aalst, 2025b)
2025-08	ADMITTED	ACP merges with A2A under the Linux Foundation (LF AI & Data Foundation, 2025)

Date	Status	Event
2025-12	ADMITTED	MCP donated to the Agentic AI Foundation (Anthropic, 2025; The Linux Foundation, 2025a)
2025-12–2026-06	CANDIDATE	recent-advances window: ∼70 preprints (Section 2.7)
2026	CANDIDATE	roadmap stage S1: generalised servers; agent traces as OCEL
2026	FORECAST	∼40% of enterprise apps feature task-specific agents (Gartner, 2025a)
2027	FORECAST	>40% of agentic-AI projects cancelled (risk controls) (Gartner, 2025c)
2028	FORECAST	∼33% of enterprise software includes agentic AI; ∼15% of work decisions autonomous (Gartner, 2025c)
2030	FORECAST	guardian agents reach ∼10–15% of the agentic-AI market (Gartner, 2025b)
2030	OPEN	roadmap stage S5: conformance-for-all-language as a default stratum (Table 9.1)

The shape of the table is the thesis in one figure: a decade of standardisation (the latent-heat plateau, Fig. 1.1), a dense recent front of research, and a forecast horizon whose realisation is, by construction, not yet ADMITTED.