ETHICAL DISCLOSURE / RED-TEAM DEMONSTRATION

THIS DOCUMENT IS A DEMONSTRATION ARTIFACT, NOT A REAL THESIS STUDY

Important: Sections labeled SIMULATED, SYNTHETIC, or NOT EXECUTED contain simulated results, predicted patterns, or unexecuted methods. They are included to demonstrate how rapidly an AI system can produce a "PhD-shaped" document and to provoke discussion about vulnerabilities in academic evaluation and credentialing. **No claim is made that experiments were executed or that results were empirically obtained.** This artifact is intended for critical discussion and research ethics discourse, not credential fraud.

[Does there exist a universal prompt grammar whose compositional structure transfers across domains and model architectures, and if so, what are its invariants?]

[whisprer]

[Computer SCience Dept., University of 6BPCF454EAUK]

[FEB 2026]

Contents

List of Figures

List of Tables

.1 Locked Thesis Spine

.1.1 Primary research question (title page + abstract)

Does there exist a universal prompt grammar whose compositional structure transfers across domains and model architectures, and if so, what are its invariants?

.1.2 Supporting lens (methods + metrics motivation)

How can we formalize prompt engineering as an information-theoretic control process over latent model trajectories, and derive prompt structures that maximize task-relevant information while minimizing hallucination-related uncertainty across model scales?

.1.3 Relationship between the two framings

The grammar/invariants question is the *what*; the information-theoretic control lens is the *why/how* that motivates metrics, causal tests, and the compiler.

.1.4 Deliverable definition (stable under negative results)

Deliverable: a formal prompt-grammar framework (operators, equivalence classes, invariants) plus a reproducible evaluation harness and a compiler design that can be empirically tested; the thesis remains valid even under negative results because boundary conditions and non-invariance regions are themselves informative outputs of the framework.

Appendix A

Introduction

A.1 Motivation: brittle heuristics and a missing theory

Prompt engineering has become a dominant interface for controlling large language models (LLMs) in practice. Yet much of the current practice remains heuristic: small changes in wording, ordering, or formatting can lead to disproportionately large changes in output behavior. This brittleness complicates reuse, comparison, and transfer of prompts across domains and model families. It also makes it difficult to determine whether an apparent improvement is a structural property of prompting, a benchmark artifact, or an accidental interaction with a specific model and decoding regime.

Recent evaluation work reinforces the need for more rigorous methodology. Robustness-focused evaluation libraries provide systematic perturbations (e.g., paraphrase, distractors, instruction injection) and show that prompt behavior can be highly sensitive under such shifts [?]. Holistic evaluation frameworks emphasize that improvements should be assessed across multiple desiderata (accuracy, robustness, calibration, efficiency, etc.), rather than as a single score [?]. Together, these observations motivate a structural approach to prompting: prompts should be treated as compositional artifacts whose effects can be studied, tested for transfer, and evaluated under controlled interventions.

A.2 Thesis spine: primary question and supporting lens

This thesis is anchored by a structural research question and supported by an information-theoretic lens.

A.2.1 Primary research question

> **Does there exist a universal prompt grammar whose compositional structure transfers across domains and model architectures, and if so, what are its invariants?**

This question treats prompts as structured objects (not merely strings) and asks whether there exist compositional regularities that persist across shifts in domain, task family, and model scale or architecture.

A.2.2 Supporting lens

A second framing motivates definitions, metrics, and experimental design:

> **How can we formalize prompt engineering as an information-theoretic control process over latent model trajectories, and derive prompt structures that maximize task-relevant information while minimizing hallucination-related uncertainty across model scales?**

In this lens, prompts act as control inputs shaping a trajectory distribution. "Better prompts" increase task-relevant signal and reduce spurious uncertainty. This lens is not treated as a claim that a single objective fully characterizes prompting, but as a unifying way to select metrics and design tests for invariance and robustness.

A.3 Contributions and deliverables

The thesis is designed so that its deliverables remain meaningful even under negative results. Specifically, the thesis aims to produce:

1. **A formal prompt-grammar framework:** prompts are represented as compositions of functional operators (role, goal, constraints, exemplars, schema, verifier, tools) and generated/parsed by a grammar. This creates explicit objects for comparison across tasks and models.

2. **Operational definitions of transfer and invariance:** invariants are defined as effects of structural features that persist across perturbation families, domains, and model variations, with explicit boundary conditions when invariance fails.

3. **A reproducible evaluation harness:** a task suite and logging pipeline supporting multi-metric, distributional evaluation across seeds and perturbations, aligned with contemporary robustness practice [?] and multi-metric evaluation principles [?].

4. **A grammar induction and testing pipeline:** methods for learning structural regularities from prompt–outcome data and testing transfer across domains and model scales.

5. **A prompt compiler prototype (design + evaluation protocol):** a synthesis/repair system that uses the grammar and discovered invariants to generate prompts predictably, and compares against baseline heuristics and optimization baselines.

Deliverable stability under failure: If universal invariants do not exist, the thesis remains successful by mapping *where* invariants fail and identifying boundary conditions. Negative results constrain the hypothesis space and are treated as first-class outcomes rather than dead ends.

A.4 Approach overview

The work proceeds in four layers:

1. **Representation:** define an operator-based ontology of prompt blocks and represent prompts as compositions (programs/parse trees).

2. **Formalization:** define perturbations, equivalence classes, invariants, and evaluation metrics (performance, validity, robustness, entropy/variance, hallucination proxies).

3. **Empirical tests:** evaluate prompt structures across tasks and models under deterministic and sampled decoding, measuring distributions over seeds and perturbations.

4. **Synthesis:** use induced structure and invariant candidates to build a prompt compiler, then evaluate it against baselines.

This approach is guided by the idea that prompt structure provides a more transferable unit of analysis than surface wording, and by ICL theory which motivates treating prompts as evidence about latent task variables [?]. Surveys and taxonomies supply vocabulary and coverage over known prompt techniques, helping define a candidate operator set [?].

A.5 Evaluation plan and success criteria

Because prompt behavior is sensitive and easy to overfit, the thesis adopts explicit success criteria:

1. **Structural predictiveness:** structural features $\phi(P)$ should predict outcomes beyond surface-text baselines on held-out tasks or models.
2. **Transfer:** learned structural regularities should generalize to at least one unseen domain and one unseen model scale or family.
3. **Robustness:** claimed improvements should persist under perturbations and across multiple decoding seeds; results must be reported distributionally rather than as single scores.
4. **Causal support:** candidate invariants should remain beneficial under controlled interventions that separate structure from wording.
5. **Practical utility:** a compiler based on learned structure/invariants should yield measurable gains relative to baseline heuristics under the same evaluation protocol.

A.6 Ethical disclosure (artifact provenance)

Ethical note: If this document is presented as a demonstration artifact rather than as an executed empirical thesis, sections labeled SIMULATED, SYNTHETIC, or NOT EXECUTED should be interpreted accordingly. This thesis template is compatible with both a fully executed empirical program and an explicitly labeled red-team demonstration intended to critique evaluation systems. In all cases, empirical claims must be supported by verifiable execution artifacts (code, logs, datasets, and reproducibility notes).

A.7 Thesis outline

Chapter **??** introduces conceptual foundations: prompting as conditioning and in-context learning, compositionality and grammars, and definitions of transfer and invariance. Chapter B surveys related work across prompt taxonomies, ICL theory, program/grammar parallels, and robustness evaluation. Chapter C presents the formal framework: prompt operators and grammars, equivalence classes, invariants, and evaluation metrics. Chapter D describes the dataset and experimental harness, including reproducibility controls. Chapter E details prompt representation and parsing. Chapter F presents grammar induction experiments and transfer tests (or, in a demonstration artifact, the planned analyses). Chapter G reports invariant candidates and intervention tests (or planned causal designs). Chapter H presents the prompt compiler design and evaluation protocol. Chapter I discusses implications, limitations, and what “universal” should and should not mean. Chapter J concludes with future work.

A.8 Purpose and scope

This thesis asks whether prompt engineering can be formalized as an information-theoretic control process: prompts act as control inputs that shape a model’s trajectory distribution during decoding, and “good” prompt structures are those that increase task-relevant information while reducing spurious uncertainty (hallucination) across model scales. The purpose of this chapter is to position that claim within prior work across four themes: (i) prompt patterns and compositional prompting, (ii) in-context learning (ICL) theory, (iii) program and grammar parallels, and (iv) robustness and causal evaluation methodology. Together, these strands motivate treating prompts not as ad hoc strings but as structured, composable objects whose effects can be measured, optimized, and tested for invariance.

A.9 Prompting as empirical control: strategies and compositional prompting

Prompting research initially emphasized the surprising ability of large language models (LLMs) to perform tasks from natural language instructions and a small number of in-context examples. Over time, the field converged on a set of repeatable “moves” that systematically improve outcomes. These moves can be interpreted as control primitives that steer generation by reshaping conditional distributions over trajectories.

A.9.1 Instruction prompting and few-shot conditioning

Instruction and few-shot prompting operate by providing a task description and, optionally, demonstrations in the prompt context. The demonstrations function as implicit training data: they convey a pattern to imitate and supply an inductive bias about the mapping from inputs to outputs. Although simple, few-shot prompting is a canonical example of control via conditioning: it modifies the model’s posterior behavior without parameter updates.

A.9.2 Decomposition and trajectory scaffolding

A major advance came from prompts that request intermediate reasoning steps (chain-of-thought prompting) or multi-stage decomposition. In control terms, these techniques introduce structured intermediate states that constrain the trajectory space: rather than directly sampling an answer, the model samples a reasoning path and then an answer conditioned on that path. Even when intermediate reasoning is not formally verified, the pattern improves accuracy on multi-step tasks, suggesting that prompting can change not only outputs but also the computation performed before committing to a final token sequence [**?**].

A.9.3 Stochastic decoding and posterior aggregation

Single-shot decoding can be a poor estimator of correctness. Self-consistency samples multiple reasoning trajectories and aggregates final answers by consensus, approximating marginalization over trajectories and using agreement as a proxy for posterior mass [**?**]. This motivates measuring not only mean performance but also variance, disagreement rates, and entropy over outputs.

A.9.4 Closed-loop prompting with tool use and observation

ReAct-style prompting interleaves reasoning with actions (e.g., retrieval) and conditions subsequent generation on observations. This resembles feedback control: intermediate state is corrected using external signals, reducing reliance on purely internal generation. Such frameworks are especially relevant to hallucination because they enable grounding—answers can be conditioned on retrieved evidence rather than latent associations [**?**].

A.9.5 Prompt chaining and modular control graphs

Prompt chaining structures complex tasks as multi-step pipelines where the output of one step becomes input to the next. AI Chains formalizes chaining with LLM primitives and demonstrates benefits for transparency and controllability in human–AI workflows [**?**]. Chaining supports the thesis framing by making prompt structure explicit as a graph of operator compositions rather than a single string.

A.9.6 Structured outputs and constrained decoding

A practically important subset of prompt engineering concerns structured outputs: JSON schemas, DSLs, SQL, and other formally constrained formats. Here, prompt structure is often paired with constrained decoding to restrict the support of the output distribution. This is an explicit entropy-control mechanism: invalid outputs are excluded by construction. However, restricting support can introduce tradeoffs: syntactic validity can improve while semantic correctness may degrade if constraints prevent the model from expressing uncertainty or nuance. Tooling ecosystems such as Outlines embody this constraint-first approach in modern structured generation pipelines [**?**].

A.10 Prompt patterns and taxonomies: vocabulary without a unifying objective

Prompt pattern catalogs and surveys provide a shared vocabulary for describing prompts as compositions of functional blocks: role assignment, goal statements, constraints, exemplars, output schemas, and self-verification instructions. These resources are valuable because they identify recurring structures and provide candidate variables for systematic study. However, this literature is largely descriptive: it explains what kinds of prompts exist rather than why particular structures work, and it rarely specifies causal mechanisms or transferable invariants. The Prompt Report synthesizes a wide range of techniques and helps standardize terminology across the community [**?**].

For this thesis, the key value of taxonomy work is to motivate a representation of prompts as compositional objects. If prompts can be decomposed into a small set of functional blocks, then the core questions become: which blocks (or compositions) drive improvements; whether their effects are stable across tasks and model families; and how they change uncertainty and hallucination risk.

A.11 In-context learning theory: prompts as inference and implicit learning

Prompting is closely tied to in-context learning: the ability of a model to adapt behavior from examples in its context window without parameter updates. A growing theory literature proposes that ICL can be understood as implicit inference over latent task variables.

A.11.1 ICL as implicit Bayesian inference

One influential account frames ICL as approximate Bayesian inference: the model uses the prompt (instructions plus examples) as evidence about a latent concept, rule, or mapping, and conditions predictions on an inferred posterior over that latent variable [?]. This perspective is crucial for an information-theoretic formalization because it introduces a latent object Z representing task-relevant structure (e.g., label mapping, format constraints, domain assumptions). If prompting causes the model to infer Z, then "good prompts" convey maximal information about Z and suppress irrelevant uncertainty.

A.11.2 Partial validity, scaling effects, and regime dependence

More recent work examines how far the Bayesian framing explains transformer ICL and where it breaks. The match can depend on model scale, training distribution, and prompt form, implying that invariants may exist only within certain regimes. Consequently, scaling is not an afterthought but a primary axis of evaluation in this thesis.

A.12 Program and grammar parallels: prompts as programs, not strings

A parallel ecosystem treats prompting as a form of programming. Prompt-programming languages and constrained generation frameworks formalize prompts as executable artifacts: they combine natural language with control flow, constraints, and structured templates.

A.12.1 Prompt-as-program languages

LMQL embeds prompts in a query/programming environment with constraints and control flow, reframing prompt engineering as designing a small program that orchestrates model calls and constrains outputs [?]. This conceptual shift matters: once prompting is programmable, prompt structure becomes explicit and can be optimized and compared across tasks and models.

A.12.2 Optimization and compilation toward "optimal structures"

Automatic Prompt Engineer (APE) frames instruction generation as search/selection over candidate prompts to maximize a score function [?]. DSPy generalizes this idea by compiling declarative LM pipelines into optimized prompting and demonstration strategies to maximize a target metric [?]. These systems are practical bridges from heuristic prompt engineering to objective-driven design, but they also expose a missing theoretical piece: optimization can find effective prompts without explaining why a structure transfers across tasks or scales, nor which properties are invariant.

A.13 Robustness and causal evaluation: prompt sensitivity and intervention tests

A persistent problem in prompt engineering research is sensitivity: small prompt changes can cause large outcome changes. This undermines naive claims of improvement and makes it easy to overfit to a benchmark through prompt tuning.

A.13.1 Prompt robustness benchmarks and perturbation testing

PromptBench provides a unified evaluation library for LLMs that includes adversarial prompt attacks, dynamic evaluation protocols, and analysis tools [?]. Such work establishes that robustness must be evaluated under systematic perturbations (e.g., paraphrase, injection, distractors, block reordering), and it provides stress tests for invariance claims.

A.13.2 Holistic multi-metric evaluation

HELM emphasizes that improvements should be judged across multiple desiderata (accuracy, calibration, robustness, efficiency, etc.) rather than single-metric optimization [?]. For this thesis, multi-metric evaluation is not only best practice; it is required by the information-theoretic framing, which implies joint objectives and explicit tradeoff reporting.

A.13.3 Toward causal claims via controlled prompt interventions

Descriptive ablations are insufficient to claim invariants. To assert that a structural block causes an improvement, evaluations should use counterfactual prompt edits: hold wording fixed while changing structure; hold structure fixed while paraphrasing; and evaluate across seeds and models. Distributional reporting—means plus variance, worst-case performance under perturbations, and entropy-like measures—reduces cherry-picked prompt success and aligns with the trajectory-distribution viewpoint advanced in this thesis.

A.14 Synthesis: what is known, what is missing, and how this thesis advances the state of the art

Prior work establishes that prompting can improve performance through reusable strategies; that ICL can be partially explained as inference over latent task variables; that prompt programming and constrained decoding make structure explicit and enforceable; and that robust evaluation requires perturbation testing and distributional reporting. However, the field lacks a unifying formal framework that links prompt structure to measurable information flow and uncertainty control across model scales.

This thesis addresses that gap by (i) representing prompts as compositions of functional operators, (ii) modeling prompting as control over trajectory distributions, (iii) defining objective functions grounded in task-relevant information and hallucination-related uncertainty, and (iv) empirically testing invariants across tasks, perturbations, and model scales using standardized evaluation harnesses.

Appendix B

Related Work

In the next chapter, these concepts are grounded in the related work literature: prompting techniques can be interpreted as control primitives over trajectories; ICL theory supplies latent-variable abstractions; prompt-as-program ecosystems make structure explicit; and robustness frameworks motivate rigorous evaluation protocols.

B.1 Overview

This chapter surveys prior work most relevant to modeling prompt engineering as structured control over model behavior. It is organized into four themes: (i) prompt patterns and taxonomies, (ii) in-context learning (ICL) theory, (iii) grammar induction and program induction parallels, and (iv) robustness and causal evaluation.

B.2 Prompt patterns and taxonomies

A substantial body of work seeks to catalog prompt engineering practices into reusable patterns and taxonomies. These resources provide shared terminology for describing prompts as compositions of functional blocks (e.g., role, goal, constraints, exemplars, output schema, verification). The Prompt Report offers a systematic survey of prompt engineering techniques and helps stabilize the vocabulary used across communities and application domains [**?**]. Although primarily descriptive, taxonomies are valuable for this thesis because they provide candidate structural variables: they suggest a space of prompt components and compositional motifs that can be treated as operators in a formal prompt grammar.

B.3 In-context learning theory

In-context learning is the conceptual backbone of many prompt engineering techniques. Theories of ICL frequently treat demonstrations and instructions as evidence about latent task structure. An influential account explains ICL as implicit Bayesian inference: prompts induce an inferred posterior over latent concepts or rules, which then governs prediction [?]. This literature supports the thesis's objective-driven framing: if a prompt conveys task-relevant information about an underlying Z, then prompt structure can be evaluated by its effectiveness at concentrating posterior mass on correct hypotheses while avoiding spurious uncertainty.

B.4 Grammar induction and program induction parallels

A parallel research ecosystem treats prompting as programming. Prompt-as-program languages and constrained generation frameworks make prompt structure explicit and executable, enabling formal constraints, control flow, and structured output guarantees.

B.4.1 Prompting for reasoning and multi-step control

Chain-of-thought prompting shows that explicitly eliciting intermediate reasoning steps can improve multi-step reasoning performance, suggesting that prompts can reshape the trajectory distribution rather than merely the final output [?]. Self-consistency further operationalizes a distributional viewpoint by sampling multiple reasoning trajectories and aggregating answers, which often improves accuracy and reduces brittleness relative to single-shot decoding [?]. ReAct interleaves reasoning and acting, using observations from tools or environments as feedback; this is naturally interpreted as closed-loop control and is especially relevant to hallucination mitigation through grounding [?]. Prompt chaining frameworks such as AI Chains describe how to compose prompts into transparent, controllable multi-step workflows, supporting the view that prompt structure can be a graph of operators rather than a single instruction [?].

B.4.2 Prompt-as-program systems and constrained decoding

LMQL formalizes prompting as a query/programming language with constraints and control flow, reframing prompt engineering as program design [?]. In structured generation settings, constrained decoding systems restrict the support of the model's output distribution to sequences that satisfy a schema or grammar, improving syntactic validity by construction. Outlines is a widely used open-source system embodying this approach for structured outputs [?]. These systems are important for this thesis because they provide practical mechanisms for enforcing structural constraints and for separating format validity from semantic correctness.

B.4.3 Optimization and compilation of prompts

Several approaches treat prompt discovery as optimization. Automatic Prompt Engineer frames prompt construction as search and selection over candidate prompts to maximize performance [?]. DSPy generalizes prompt optimization into compilation of declarative LM pipelines into improved prompting and demonstration strategies [?]. These systems demonstrate that prompt structures can be optimized, but they typically do not identify which properties of the resulting prompts are invariant across domains or architectures. This gap motivates the present thesis's focus on structural invariants and equivalence classes.

B.5 Robustness and causal evaluation

Prompt sensitivity is a central obstacle to reliable prompt engineering: small perturbations to wording or structure can produce large changes in output. Consequently, robust evaluation must examine distributions over prompt variants, decoding seeds, and adversarial perturbations rather than single prompts.

PromptBench provides a unified evaluation library with perturbation and attack suites for assessing robustness of LLM behaviors under adversarial and distribution-shifted prompt conditions [?]. HELM emphasizes holistic, multi-metric evaluation across scenarios and desiderata, highlighting that improvements in one metric can trade off against others [?]. Together, this literature motivates the evaluation protocols used in this thesis: controlled interventions (structure fixed vs wording varied; wording fixed vs structure altered), distributional reporting (means, variances, worst-case outcomes), and cross-model testing to support or refute invariance claims.

B.6 Summary

Prior work provides (i) taxonomies that identify candidate prompt structures, (ii) ICL theory that motivates latent-variable interpretations of prompting, (iii) programmatic prompting and constrained decoding systems that make structure explicit and enforceable, and (iv) robustness frameworks that clarify why naive prompt comparisons are insufficient. The next chapter builds on these foundations to define a formal prompt grammar, equivalence classes, invariants, and evaluation metrics aligned with an information-theoretic control perspective. Patch to your master thesis.tex (to include these two)

Appendix C

Formal Framework

C.1 Purpose

This chapter defines the formal objects used throughout the thesis: prompts as compositional artifacts, model decoding as trajectory distributions, and invariants as structure-dependent effects that persist under perturbation and shift. The definitions are written to support the experimental design, not to overfit to any single model family.

C.2 Notation and basic objects

Let $x \in \mathcal{X}$ denote a task input and $y \in \mathcal{Y}$ denote an output sequence. Let θ denote fixed model parameters. A *decoding policy* $d \in \mathcal{D}$ specifies sampling/decoding parameters (e.g., temperature, nucleus threshold, maximum tokens). A *prompt* P is treated as an artifact presented alongside x.

We emphasize trajectory-level analysis. Let $\tau = (y_1, \dots, y_T)$ denote the full token trajectory. The model induces a trajectory distribution:

$$p_\theta(\tau \mid x, P, d). \tag{C.1}$$

The corresponding marginal over final outputs is $p_\theta(y \mid x, P, d)$.

C.3 Prompt operators and compositional structure

We represent prompts using functional blocks (role statements, constraints, exemplars, schemas, verifiers) and treat prompt construction as composition of *operators*.

Definition C.1 (Prompt state). *A* prompt state *is a structured object S containing (at minimum) a text field plus optional structured metadata (e.g., a schema, constraint set, or tool policy).*

Definition C.2 (Prompt operator). *A* prompt operator *is a (possibly context-dependent) map*

$$\mathcal{O}_i : (S, x) \mapsto S',$$

that transforms a prompt state by injecting a functional block (e.g., prepend role, append schema, insert exemplars, add verifier checklist).

Definition C.3 (Prompt program / composition). *A* prompt program *(or* prompt structure*) is a finite composition*

$$\Pi = \mathcal{O}_k \circ \cdots \circ \mathcal{O}_2 \circ \mathcal{O}_1,$$

producing S_k from an initial S_0. The realized prompt text is $\text{text}(S_k)$.

C.4 Perturbations, shifts, and equivalence

We distinguish three kinds of changes: (i) *surface* changes (paraphrase), (ii) *structural* changes (operator reorder/remove/replace), and (iii) *distribution shifts* (domain shift, model family/scale changes).

Definition C.4 (Perturbation family). *A* perturbation π *is a transformation of prompt text and/or structure. A* perturbation family Π *is a set of such transformations (e.g., paraphrase, injection, distractors, block reordering).*

Definition C.5 (Structural equivalence class). *Two prompts P and P' are* structurally equivalent *(written $P \sim P'$) if they share the same prompt program Π (operator sequence/tree) up to an allowed set of structure-preserving transformations (e.g., paraphrase within blocks). The equivalence class is $[P] = \{P' : P' \sim P\}$.*

C.5 Invariants

Definition C.6 (Effect of a structural feature). *Let f be a measurable outcome metric (accuracy, validity, robustness score, etc.). The* effect *of adding a structural feature $\mathcal{O}$ in context (x, m, d) is the difference:*

$$\Delta_{\mathcal{O}}(f) = \mathbb{E}[f \mid P \oplus \mathcal{O}] - \mathbb{E}[f \mid P],$$

where expectations are taken over sampling seeds / trajectories as defined by d.

Definition C.7 (Invariant (operational))**.** *A structural feature $\mathcal{O}$ is an* invariant *for a metric f over a set of conditions $\mathcal{C}$ (domains, models, decoders, perturbations) if:*

$$\Delta_{\mathcal{O}}(f; c) \geq \epsilon \quad \forall c \in \mathcal{C},$$

for some $\epsilon > 0$, and the effect remains non-negative under the perturbation family Π.

C.6 Evaluation metrics

We evaluate prompts along multiple axes:

- **Task performance:** accuracy/F1/EM or task-specific score.
- **Format validity:** schema/grammar validity rate for structured tasks.
- **Robustness:** worst-case drop under perturbations; stability across seeds.
- **Entropy / variance proxies:** output disagreement or categorical entropy across sampled trajectories.
- **Hallucination proxies:** unsupported claim rate or abstention correctness (task-dependent).

C.7 Hypotheses

H1 (Structure improves reliability). Adding constraint and verification operators (e.g., Constraints, Verifier, Abstain) improves mean task score and reduces variance across seeds relative to baseline prompts.

H2 (Hard constraints reduce syntactic entropy). Enforcing schemas/grammars at decode time increases format validity and reduces entropy of invalid outputs, with potential semantic tradeoffs.

H3 (Closed-loop control reduces hallucination). Tool-mediated feedback (retrieve/observe loops) reduces hallucination proxies on knowledge-sensitive tasks compared to open-loop prompting.

H4 (Some effects are scale-dependent). The magnitude and sign of $\Delta_{\mathcal{O}}(f)$ may vary with model scale and family, implying boundary conditions on invariants.

C.8 Chapter 4 Freeze: "TODO List with Math"

The following list is intended to be **frozen** as the Chapter 4 backbone. It should not change even if experiments yield negative results.

C.8.1 4.1 Objects and notation (Definitions)

D1. LLM as conditional trajectory distribution $p_\theta(\tau \mid x, P, d)$.

D2. Prompt state S and realized prompt $\text{text}(S)$.

D3. Prompt operator $\mathcal{O}_i : (S, x) \mapsto S'$.

D4. Prompt program/structure $\Pi = \mathcal{O}_k \circ \cdots \circ \mathcal{O}_1$.

D5. Block ontology (Role/Goal/Constraints/Schema/Exemplars/Verifier/Tools/Abstain).

C.8.2 4.2 Grammar and parsing (Definitions)

D6. Prompt grammar G generating operator sequences/trees.

D7. Parse tree / derivation for a prompt instance.

D8. Structural feature map $\phi(P)$ (presence/order/arity/length, etc.).

C.8.3 4.3 Perturbations, equivalence, invariance (Definitions)

D9. Perturbation family Π (paraphrase, reorder, distractor, injection).

D10. Structural equivalence $P \sim P'$ and equivalence class $[P]$.

D11. Effect size $\Delta_{\mathcal{O}}(f)$ for metric f.

D12. Operational invariant (minimum effect ϵ over condition set $\mathcal{C}$).

C.8.4 4.4 Metrics (Definitions)

D13. Task score f_{task} (accuracy/F1/EM).

D14. Validity f_{valid} (schema/grammar validity).

D15. Robustness f_{rob} (worst-case drop under Π).

D16. Entropy/variance proxy f_{ent} (seed entropy/disagreement).

D17. Hallucination proxy f_{hall} (unsupported claim rate / abstain correctness).

C.8.5 4.5 Propositions (lightweight; not dependent on experiments)

P1. Support restriction reduces syntactic entropy. Hard constraints restrict support, therefore reduce invalid-format entropy by construction.

P2. Variance decomposition. Total variability can be decomposed into seed variance + prompt-variant variance + perturbation variance (operational decomposition used in reporting).

P3. Structure vs wording identifiability (operational). Under controlled interventions (wording-only vs structure-only edits), we can attribute changes to structure up to residual confounds (a methodological proposition, not a theorem).

C.8.6 4.6 Hypotheses (testable; allowed to be falsified)

H1. Constraints+Abstain increase reliability and reduce hallucination proxies.

H2. Verifier reduces seed entropy / increases stability.

H3. Hard schema constraints increase validity with a possible semantic tradeoff.

H4. Closed-loop prompting reduces hallucination proxies on grounded tasks.

H5. Some operator effects are scale-dependent; invariants have boundary conditions.

Lock statement: The items in Sections C.8.1–C.8.6 constitute the Chapter 4 "math TODO list" and are intended to remain stable even if subsequent empirical findings are negative or mixed.

Appendix D

Dataset and Experimental Harness

D.1 Scope and disclosure

Disclosure: This chapter specifies an evaluation harness and datasets. Execution is described as a plan. Where synthetic results are discussed elsewhere, they are explicitly labeled SIMULATED. No empirical claims are made here.

D.2 Task suite

The harness is designed to cover structured output, reasoning, and hallucination pressure. A canonical suite includes:

- **AG News** (classification; JSON output schema)
- **GSM8K** (math reasoning; exact match on final answer)
- **TruthfulQA** (truthfulness/hallucination pressure; abstention permitted)
- **CoNLL-2003** (NER extraction; span-F1 with JSON entities)
- **CommonsenseQA** (MCQ; structured choice output)
- **FEVER** (claim verification; label accuracy; optional evidence)

D.3 Model grid and scale axis

Evaluation is intended across a scale grid (small/medium/large) and, where feasible, across model families. The goal is not to crown a best model but to observe whether structural effects persist under architectural and scaling changes.

D.4 Prompt conditions and operator bundles

Prompts are generated from operator bundles (Conditions C0–C8). Baseline

- C2: add constraints + abstain rule
- C4: add verifier checklist
- C6: hard schema constraint at decode time
- C7: closed-loop ReAct (retrieve–observe-answer)
- C8: optimized/compiled (e.g. DSPy) *when used as a baseline*

D.5 Decoding regimes and variance handling

Two regimes are used:

- D0: deterministic decoding (temperature = 0)
- D1: sampled decoding (e.g., temperature = 0.7, top-$p = 0.95$), repeated across R seeds

Distributional reporting (mean, variance, worst-case) is required for robustness claims.

D.6 Logging and reproducibility

Each run records: task id, item id, model id, condition id, decoding regime, seed, prompt hash, output, parsing outcome, metrics, and errors. This supports auditability and enables red-team demonstration of how little human intervention is required to generate "thesis-shaped" artifacts.

D.7 Robustness perturbations

A perturbation family includes paraphrase, block reorder, distractors, and instruction injection. Robustness is r

Appendix E

Prompt Representation and Parsing

E.1 Block ontology

We model prompts as compositions of functional blocks:

Block	**Function**
Role	sets stance/identity and behavioral priors
Goal	states task objective and success criteria
Constraints	imposes must/should rules, refusal/abstain policies
Schema	specifies output language (JSON/DSL/format)
Exemplars	provides demonstrations (few-shot)
Verifier	induces self-check and revision behavior
Tools	specifies tool use policy (retrieve/compute)

E.2 Operator-based representation

Each block corresponds to an operator $\mathcal{O}_i$ acting on prompt state S. This representation separates: (i) structure (operator sequence/tree) from (ii) realization (actual token text of blocks).

E.3 Parsing and annotation

Two parsing modes are supported:

- **Constructive parsing:** prompts are generated from known operator bundles (C0–C8), so their structure is known by construction.

- **Retrospective parsing:** natural prompts are segmented into blocks using rules and/or annotators.

E.4 Inter-annotator agreement (if used)

If human annotation is employed, agreement is evaluated with standard measures (e.g., Cohen's κ) at the block-boundary level and at block-type labeling level.

E.5 Baseline structural features

Baseline features include: presence/absence of blocks, block ordering, block lengths, constraint count, exemplar co

(explicitly NOT executed; includes SIMULATED outputs later in appendix)

Appendix F

Grammar Induction Results (Planned) and Expected Patterns

F.1 Disclosure: not executed

NOT EXECUTED: The empirical induction described here is a plan and a set of predicted patterns. Any numerical tables or plots referenced as examples are provided only in Appendix D and are labeled SIMULATED.

F.2 Induction goal

We aim to learn whether prompt structure predicts outcomes and whether learned structures generalize across domains and models. Concretely:

- learn a mapping from structural features $\phi(P)$ to metrics f
- compare structure-based predictors against surface-text baselines
- test transfer across domains and model scales

F.3 Methods (planned)

Planned methods include:

- **Feature models:** linear/logistic regression and tree-based models on structural features.

- **Grammar models:** PCFG-like operator grammars where productions represent block compositions.
- **Program induction:** discovering recurring operator programs that maximize J (multi-metric objective).

F.4 Expected patterns (predictions)

Based on prior work and engineering intuition, we expect:

- **P1:** Schema and Hard constraint drastically increase format validity (especially on JSON tasks).
- **P2:** Constraints + Abstain reduce unsupported-content proxies on hallucination-sensitive tasks.
- **P3:** Verifier reduces variance across seeds (lower disagreement/entropy).
- **P4:** Exemplars have variable benefit: stronger on small models and brittle under perturbations.

F.5 Transfer tests (planned)

We propose train/test splits across: (i) domains (e.g., train on classification + extraction, test on verification), (ii) models (train on small/medium, test on large), (iii) perturbations (train on clean prompts, test on paraphrase/injection).

F.6 Ablations (planned)

We propose operator ablations that hold wording fixed while changing structure (remove Verifier; reorder Constraints and Schema; replace Hard constraint with post-hoc validation).

Appendix G

Invariants and Causal Tests (Planned)

G.1 Disclosure: planned causal design

NOT EXECUTED: This chapter specifies causal-style intervention tests and predicted boundary conditions. Demonstration tables/figures are provided as SIMULATED in Appendix D.

G.2 Invariant candidates

Candidate invariants include:

- **I1: Schema+constraint improves format reliability** across tasks/models.
- **I2: Verifier reduces seed variance** (lower entropy in categorical outputs).
- **I3: Abstain reduces hallucination proxies** when questions are underdetermined.
- **I4: Hard constraints outperform retry loops** for validity at similar cost.

G.3 Intervention experiments (structure vs wording)

Two principal interventions:

1. **Structure fixed, wording varied:** paraphrase within blocks while holding operator program constant.
2. **Wording fixed, structure altered:** keep much of the same text but reorder blocks or change scope (e.g., constraints apply only to output vs whole response).

An effect is considered robust if it persists across perturbations and seeds.

G.4 Boundary conditions (predictions)

We predict failures in regimes such as:

- extremely long contexts (constraint dilution),
- tasks where correctness requires creativity/ambiguity (over-constraining harms),
- model families that handle tool use differently (closed-loop effects vary),
- heavy instruction injection (security boundary is not stable).

G.5 Negative results as first-class outcomes

A core claim of this thesis is that negative results are informative: they map where invariants do *not* ex

Appendix H

The Prompt Compiler (Prototype Design)

H.1 Design goals

The prompt compiler aims to transform a task specification into a structured prompt program (operator composition) that improves reliability and robustness. It must:

- produce auditable outputs (prompt hashes, operator traces),
- support constraints (schemas/grammars),
- support tool loops when needed (closed-loop),
- be optimizable under a multi-metric objective.

H.2 Architecture

The compiler consists of:

1. **Front-end:** task descriptor $\rightarrow$ required blocks (schema, labels, evidence rules).
2. **Planner:** select operator program Π from a grammar / library.
3. **Back-end:** render prompt text + attach constraint engine (if any).
4. **Validator:** parse outputs and optionally trigger minimal repair (bounded retries).

H.3 Generation algorithm (planned)

We propose grammar-guided synthesis:

- choose a candidate operator sequence Π from a weighted grammar,
- render prompt and evaluate on a dev set,
- update weights to favor programs with better objective J.

Alternatively, use a fixed library of invariants (if discovered) as deterministic compilation rules.

H.4 Evaluation protocol (planned)

The compiler is evaluated against baselines:

- naive baseline prompt (C0),
- heuristic prompting (C2/C4),
- constrained decoding (C6),
- optimization baseline (C8, if used).

Metrics include task score, validity, robustness under perturbations, entropy/variance, and token cost.

This is where the “elephant” argument lives, ethically.

Appendix I

Discussion

I.1 What this artifact demonstrates (meta-level)

This document is intentionally “thesis-shaped” and uses modern AI assistance to demonstrate a systemic issue: many academic evaluation processes emphasize *form* (structure, citations, methodological language) in ways that can be satisfied without corresponding *epistemic substance* (executed experiments, verified claims). The purpose is not to deceive but to make visible how little effort may be required to produce credible-looking academic prose when assisted by an LLM.

I.2 Universality: what it could mean (and what it cannot)

“Universal” prompt grammar cannot mean a single structure works best for all tasks. A defensible meaning is weaker: that certain structural operators have stable effects over defined regimes, and that failures can be mapped as boundary conditions. This reframes universality as *conditional invariance* rather than absolute invariance.

I.3 Implications for academic evaluation and credentialing

The existence of plausibly formatted, citation-bearing, methodologically articulated documents produced with minimal human input suggests that credential systems must shift toward:

- stronger verification of empirical execution,
- artifact-based review (code, logs, preregistration),
- adversarial assessment (red-team evaluation of claims),
- explicit treatment of AI assistance and provenance.

I.4 Implications for reliability and safety in LLM deployment

From a technical standpoint, prompt structure as control has safety implications: injection attacks, constraint bypass, and brittle instruction hierarchies are all instances of failing invariance under perturbation. Robust prompt engineering therefore intersects with security and trustworthy AI evaluation.

I.5 Limitations of the present artifact

This document intentionally includes SIMULATED results rather than executed experiments. Its streng

Appendix J

Conclusion and Future Work

J.1 Summary

This thesis frames prompt engineering as compositional control over model trajectory distributions and proposes studying prompt structure via grammars, invariants, and causal interventions. It also serves as a red-team demonstration artifact: a highly structured academic document can be produced rapidly with AI assistance, motivating a re-examination of evaluation practices.

J.2 Future work (technical)

Future technical directions include:

- adaptive prompt grammars conditioned on task and model family,
- model-in-the-loop compilers with verified tool grounding,
- formal security models for prompt injection robustness,
- improved hallucination metrics that separate epistemic from syntactic uncertainty.

J.3 Future work (institutional)

Future institutional directions include:

- provenance standards for AI-assisted writing,
- artifact submission requirements (code, logs, preregistration),

- adversarial review procedures for empirical claims,
- updated credentialing frameworks emphasizing demonstrated capability over prose form.

Appendix A

Annotation Guidelines

A.1 Block definitions

Annotators label prompt segments as: Role, Goal, Constraints, Schema, Exemplars, Verifier, Tools, or Other.

A.2 Boundary rules

- A Schema block must include explicit formatting requirements (JSON keys/types or grammar).
- A Constraints block contains must/should rules or refusal/abstain policies.
- A Verifier block contains self-check instructions and/or revision prompts.

A.3 Disagreements

If a segment could be Constraints or Verifier, prefer Verifier if it contains explicit checki

“‘

Appendix B

Example Prompts and Parses

B.1 Example: C4 structure (Verifier)

Structure: Role → Goal → Constraints → Abstain → Verifier

```
ROLE: You are a careful, literal assistant. Follow instructions exactly.
GOAL: Classify the news topic.
CONSTRAINTS:
- Do not invent facts.
- If unsure, abstain.
SCHEMA: Output valid JSON with keys {"label","confidence","rationale"}.
VERIFIER:
- Check schema validity.
- Check constraints.
- If any check fails, revise once.
```

B.2 Red-team note (provenance)

This prompt structure was generated programmatically from operator bundles with minimal human input.

Appendix C

Code and Reproducibility Notes

C.1 Artifact logging

Each run logs prompt hashes, model ids, decoding parameters, outputs, parse success, metrics, and errors.

C.2 Ethical disclosure

All simulated results are labeled SIMULATED and generated under explicit assumptions in Appendix D. This artifact must not be cited as empirical evidence.

The Synthetic Results Appendix (the “unmissable” bit)

Appendix D

Synthetic Results Appendix (Red-Team Demonstration)

D.1 Ethical statement

SIMULATED RESULTS ONLY. This appendix contains simulated, fabricated, or model-based predicted results. They are included solely to demonstrate how easily a "PhD-shaped" results section can be produced. **These numbers do not come from executed experiments.**

D.2 Simulation assumptions

The simulation is constructed to be plausible given common empirical patterns reported in the prompt engineering literature:

- Adding explicit constraints and abstention reduces hallucination proxies but can reduce answer rate.
- Adding verifiers reduces variance across seeds.
- Hard schema constraints dramatically increase JSON validity.
- Tool feedback (closed-loop) reduces unsupported claims in verification tasks.

We therefore simulate effect sizes that are modest, not miraculous, and include variance and failure modes.

D.3 SIMULATED summary tables

Table D.1: SIMULATED AG News accuracy (mean over seeds) by condition.

Condition	Small	Medium	Large
C0 (baseline)	0.80	0.84	0.87
C2 (+constraints)	0.81	0.85	0.88
C4 (+verifier)	0.82	0.86	0.88
C6 (hard schema)	0.81	0.85	0.88

Table D.2: SIMULATED JSON validity rate on structured tasks (higher is better).

Condition	CoNLL JSON validity	FEVER JSON validity
C0	0.62	0.58
C2	0.78	0.74
C4	0.84	0.81
C6 (hard constraint)	0.99	0.99

Table D.3: SIMULATED categorical entropy (D1 sampled) for classification/verification outputs (lower is more stable).

Condition	AG News entropy	FEVER label entropy
C0	0.42	0.51
C2	0.36	0.44
C4	0.28	0.33
C6	0.30	0.35

D.4 SIMULATED figures (textual placeholders)

SIMULATED FIGURE PLACEHOLDER: In a real run, this section would include plots generated by the harness (condition ladders, robustness drops, Pareto frontiers). Here we intentionally omit actual figures and instead emphasize that producing them from simulated data is trivial.

D.5 Red-team commentary: why this is dangerous

It is straightforward to generate tables like Tables D.1–D.3, write a plausible narrative, and pass superficial review processes that prioritize formatting, citations, and methodological language. The point of this appendix is to demonstrate that such a document can look legitimate while containing no executed experiments.

D.6 Hard safety line

This artifact is constructed to *avoid* academic fraud: everything here is unmistakably labeled SIMULATED figures (textual placeholders) **SIMULATED FIGURE PLACEHOLDER:** In a real run, this section would include plots generated by the harness (condition ladders, robustness drops, Pareto frontiers). Here we intentionally omit actual figures and instead emphasize that producing them from simulated data is trivial.

D.7 Red-team commentary: why this is dangerous

It is straightforward to generate tables like Tables D.1–D.3, write a plausible narrative, and pass superficial review processes that prioritize formatting, citations, and methodological language. The point of this appendix is to demonstrate that such a document can look legitimate while containing no executed experiments.

D.8 Hard safety line

This artifact is constructed to *avoid* academic fraud: everything here is unmistakably labeled SIMULATED. Any attempt to remove labels and present these results as real would be deception and is explicitly disavowed.

...

Bibliography

‘‘‘

Bibliography

refs.bib (BibTeX key plan + starter entries)

@miscwei2022cot, title = Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, author = Wei, Jason and Wang, Xuezhi and Schuurmans, Dale and Bosma, Maarten and Ichter, Brian and Xia, Fei and Chi, Ed H. and Le, Quoc V. and Zhou, Denny, year = 2022, eprint = 2201.11903, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2201.11903

@miscwang2022selfconsistency, title = Self-Consistency Improves Chain of Thought Reasoning in Language Models, author = Wang, Xuezhi and Wei, Jason and Schuurmans, Dale and Le, Quoc V. and Chi, Ed and Narang, Sharan and Chowdhery, Aakanksha and Zhou, Denny, year = 2022, eprint = 2203.11171, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2203.11171

@inproceedingsyao2023react, title = ReAct: Synergizing Reasoning and Acting in Language Models, author = Yao, Shunyu and Zhao, Jeffrey and Yu, Dian and Du, Nan and Shafran, Izhak and Narasimhan, Karthik and Cao, Yuan, booktitle = International Conference on Learning Representations (ICLR), year = 2023, eprint = 2210.03629, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2210.03629

@miscwu2021aichains, title = AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts, author = Wu, Tongshuang and Terry, Michael and Cai, Carrie J., year = 2021, eprint = 2110.01691, archivePrefix = arXiv, primaryClass = cs.HC, url = https://arxiv.org/abs/2110.01691, note = Also appeared in CHI 2022.

@miscschulhoff2024promptreport, title = The Prompt Report: A Systematic Survey of Prompt Engineering Techniques, author = Schulhoff, Sander and Ilie, Michael and Balepur, Nishant and Kahadze, Konstantine and Liu, Amanda and Si, Chenglei and others, year = 2024, eprint = 2406.06608, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2406.0

@miscxie2021iclbayes, title = An Explanation of In-context Learning as Implicit Bayesian Inference, author = Xie, Sang Michael and Raghunathan, Aditi and Liang, Percy and Ma, Tengyu, year = 2021, eprint = 2111.02080, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2111.02080

@miscbeurerkellner2022lmql, title = Prompting Is Programming: A Query Language for Large Language Models, author = Beurer-Kellner, Luca and Fischer, Marc and Vechev, Martin, year = 2022, eprint = 2212.06094, archivePrefix = arXiv, primaryClass = cs.PL, url = https://arxiv.org/abs/2212.06094, note = Associated with PLDI 2023 artifact track; LMQL implementation available online.

@misczhou2022ape, title = Large Language Models Are Human-Level Prompt Engineers, author = Zhou, Yongchao and Muresanu, Andrei Ioan and Han, Ziwen and Paster, Keiran and Pitis, Silviu and Chan, Harris and Ba, Jimmy, year = 2022, eprint = 2211.01910, archivePrefix = arXiv, primaryClass = cs.LG, url = https://arxiv.org/abs/2211.01910

@misckhattab2023dspy, title = DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, author = Khattab, Omar and Singhvi, Arnav and Maheshwari, Paridhi and Zhang, Zhiyuan and Santhanam, Keshav and Vardhamanan, Sri and Haq, Saiful and Sharma, Ashutosh and Joshi, Thomas T. and Moazam, Hanna and Miller, Heather and Zaharia, Matei and Potts, Christopher, year = 2023, eprint = 2310.03714, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2310.03714

@articlezhu2024promptbench, title = A Unified Library for Evaluation of Large Language Models, author = Zhu, Ke and others, journal = Journal of Machine Learning Research, year = 2024, volume = 25, number = 23, url = https://jmlr.org/papers/v25/24-0023.html, note = Introduces PromptBench; arXiv version: 2312.07910.

@miscliang2022helm, title = Holistic Evaluation of Language Models, author = Liang, Percy and Bommasani, Rishi and Lee, Tony and Tsipras, Dimitris and Soylu, Dilara and Yasunaga, Michihiro and Zhang, Yian and Narayanan, Deepak and Wu, Yuhuai and others, year = 2022, eprint = 2211.09110, archivePrefix = arXiv, primaryClass = cs.CL, url = https://arxiv.org/abs/2211.09110

@miscoutlines2024repo, title = Outlines: Structured Outputs, author = dottxt-ai, year = 2024, howpublished = `https://github.com/dottxt-ai/outlines`, note = Software repository (accessed 2026-02-22).