# Fidelity Is Not Effectiveness: Limits of the Mutation Score in LLM-Inferred Mealy Machines

Alecsander G. de Matos
ICMC/USP
São Carlos, Brazil
alecsander.matos@usp.br

Simone R. S. Souza
ICMC/USP
São Carlos, Brazil
srocio@icmc.usp.br

Márcio Eduardo Delamaro
ICMC/USP
São Carlos, Brazil
delamaro@icmc.usp.br

## Abstract

Model-based testing (MBT) derives test cases from behavioral models, whose construction may require manual effort and specialized knowledge. Large language models (LLMs) offer an alternative by inferring such models directly from natural-language requirements, but the resulting test suites are trustworthy only if the inferred model faithfully represents the expected behavior. We investigate whether the mutation effectiveness of input sequences generated from an inferred Mealy machine provides sufficient evidence of behavioral fidelity. In a retrospective analysis of 30 pipeline runs, all final models were valid and minimal, but two were behaviorally non-equivalent to the reference even though their test suites achieved the same mutation score as a suite generated from the reference model. We prospectively replicated the experiment with 50 runs under a predefined protocol and a fixed LLM version. Seven runs again produced non-equivalent models whose suites achieved the same score as the reference-derived suite across three classical FSM test-generation methods. The result persists after excluding mutants that make the machines partial and does not arise because the two test suites are identical. A contrasting case shows that mutation score can reveal a loss of test effectiveness when model errors alter the selected input sequences. Therefore, mutation effectiveness is useful as a complementary measure, but equality with the reference-derived score does not certify behavioral fidelity. Formal validity, behavioral equivalence, and test-suite effectiveness should be evaluated separately.

## Keywords

Model-Based Testing, Mealy Machines, Large Language Models, Model Inference, Behavioral Equivalence, Behavioral Mutation, Fidelity

## 1 Introduction

Model-based testing (MBT) derives test cases from behavioral models of the system or of its environment [16]. In state-machine-based conformance testing, classical methods offer fault-detection guarantees under explicit hypotheses about the model, the implementation, and the fault domain [12]. In practice, however, obtaining the model remains a costly step: it is usually built manually from natural-language requirements and depends on specialized knowledge.

Because these requirements are expressed in natural language, large language models (LLMs) can infer the model itself, and not only the tests, directly from them [1, 15, 22]. This possibility may reduce part of the modeling effort, but it also introduces a verification step: before trusting the generated suites, one must assess whether the inferred model correctly represents the expected behavior.

This assessment requires distinguishing formal validity from behavioral fidelity. A machine can be complete and deterministic, with every state reachable, and still produce outputs that differ from those prescribed by the reference. In that case, it is formally eligible to generate suites, yet it remains an unfaithful model. Minimality does not solve the problem either: it eliminates behaviorally redundant states without guaranteeing that the remaining behavior is the correct one.

One possible strategy is to compare the effectiveness of the sequences generated from the inferred model with that of a suite derived from the reference model. To this end, we measure the fraction of reference mutants killed by the sequences of each suite. We call this measure the *reference-oracled input-sequence mutation score*. In this computation, however, the kill oracle is the reference itself. The metric assesses the effectiveness of the sequences against the mutant population, but it does not directly check whether the expected outputs prescribed by the inferred model are correct.

Given this limitation, we investigate whether the fact that a suite derived from an inferred model reaches the same *mutation score* as the reference suite provides sufficient evidence of behavioral fidelity. We show that it does not. A model can be non-equivalent to the reference and still select sequences that retain the same effectiveness against the mutant population. Matching the score of the reference suite therefore does not certify fidelity.

Our conclusion is one of insufficiency, not of prevalence. The goal is not to estimate how often this behavior occurs across models, systems, or LLMs, but to determine whether equality between the scores can be used as a certificate of equivalence. To refute that interpretation, it suffices to observe one situation in which a non-equivalent model produces a suite as effective as the reference suite.

The evidence comes from two complementary stages: a retrospective analysis of previously collected runs and a prospective replication with a fixed model snapshot, a protocol frozen before data collection, and a predefined analysis. We complement these stages with a contrasting case and analyses of suite overlap and operator sensitivity.

The main contributions are:

(1) An auditable evaluation pipeline that explicitly separates formal validity, behavioral fidelity, and suite effectiveness for LLM-inferred Mealy machines. The pipeline uses inference to produce the models under evaluation, but the inference procedure itself is not claimed as a methodological contribution.
(2) Empirical evidence, built in two stages, that the *reference-oracled input-sequence mutation score* is not sufficient as a proxy for fidelity. In both the retrospective analysis and

the prospective replication, valid, minimal, non-equivalent models produced suites with the same score as the reference suite. The seven prospective non-equivalent models shared the same behavioral error pattern, rather than exhibiting seven distinct classes of infidelity. The result does not stem from identical suites (Section 5.3) and persists even when the partial mutants are excluded (Section 5.4).

(3) A characterization of the situations in which infidelity remains invisible to the metric and of those in which it becomes observable through a loss of effectiveness. The analysis is accompanied by recommendations for evaluating the model and the suite separately, as well as reproducible scripts for suite comparison, operator sensitivity, and equivalence *cross-check*.

The research questions are stated in Section 4 and the results are presented in Section 5.

## 2 Background

### 2.1 Mealy Machines, Equivalence, and Minimality

A Mealy machine is a deterministic finite-state transducer that, on each input symbol, moves to a new state and emits an output symbol. We use machines that are complete (every state/input combination has a defined transition) and deterministic. Two machines are behaviorally equivalent ($M \equiv R$) when they produce the same output sequence for every input sequence. A machine is minimal when no two of its states are behaviorally equivalent; the minimal machine realizing a given behavior is unique up to state renaming. In MBT, the reference model is the oracle of the expected behavior, and conformance asks whether the observable behavior of the implementation agrees with that of the model [12, 16].

### 2.2 Input Sequences and Output Oracles

A test suite derived from a machine combines two elements: the input sequences that exercise the system and the expected outputs associated with those sequences. Building oracles, that is, deciding whether an observed output is correct, is a fundamental difficulty of software testing [4]. This distinction is central to this paper: a suite may contain sequences that adequately exercise the system and still rely on expected outputs derived from an unfaithful model. For example, suppose an inferred model generates an input sequence that exercises the same relevant behaviors as a sequence generated from the reference model, but predicts a different output for one of its inputs. The sequence may still be effective at exposing faults when the reference model supplies the kill oracle, even though the inferred model's own expected output is incorrect. Thus, sequence effectiveness and model fidelity are distinct properties.

### 2.3 Generation Methods with Guarantees: W, Wp, and HSI

Several classical test-generation methods for finite-state machines provide fault-detection guarantees. The W method derives a test set that detects any implementation, within a declared fault domain, that differs from the specification [6], building on the classical coverage bound for extra states [19]. The partial W method (Wp) retains this detection power with smaller suites [10], while HSI (*Harmonized State Identifiers*) employs harmonized identifiers [13].

The guarantees are relative to each method's hypotheses: a bound on the number of implementation states, a reliable *reset*, a complete and deterministic specification, reachability of all states, and a shared alphabet [7]. Recent work continues to extend the fault domains under which these guarantees remain valid [18]. A complete suite is thus only required to distinguish the specification from non-equivalent implementations when they belong to the considered fault domain and satisfy the method's hypotheses. Section 4.7 discusses how these conditions relate to the mutation operators used in the experiment.

### 2.4 Mutation as an Effectiveness Criterion

Mutation injects syntactic defects (mutants) and measures the fraction killed by a test suite. Studies with code mutants provide evidence that the *mutation score* can serve as a proxy for suite effectiveness in certain contexts [2, 11]. This evidence does not automatically validate behavioral mutation on Mealy machines, which we treat here as a distinct domain. In this study, mutation is applied to the behavior of Mealy machines, following the tradition of mutation analysis for FSM-based specifications [8]. The operators and their semantics are presented in Section 4.7. Behavioral mutation is not the same as code-level *mutation testing*, and mutants are not real faults.

## 3 Related Work

*LLM-based FSM inference from requirements.* Recent work generates state machines from natural-language requirements and evaluates them by criteria different from ours: mutation-guided repair and test generation, observing that the most frequent errors involve incorrect outputs and missing transitions [15]; component-level *F1-score* (states, transitions, guards, and actions) on UML machines [1]; and execution-based output matching in RTL generation, in the *hardware* domain [22]. Our study addresses a more fundamental question: whether the effectiveness of the sequences, evaluated with the reference as the oracle, suffices on its own as a proxy for *behavioral fidelity*. We show that it does not. Accordingly, the inference stage is used to produce the models under evaluation rather than proposed as a new inference technique. Our contribution is the evaluation framework and the assessment of whether sequence effectiveness can serve as sufficient evidence of behavioral fidelity.

**Test generation with LLMs.** *Surveys* map the use of LLMs in software testing [20], and there are systems that generate tests directly, some guided by mutation [9, 14]. These systems generate tests directly from code, without inferring an intermediate formal behavioral model; in this work, we instead infer and evaluate an intermediate formal behavioral model.

**MBT and completeness methods.** W, Wp, and HSI and their hypotheses are classical [6, 7, 10, 13]; we use these methods as the generation layer and their guarantees as a theoretical reference, always qualified by the hypotheses.

**Mutation as an evaluation criterion.** There is empirical evidence, from studies with code, on the relation between mutant

detection and the detection of real faults [2, 11]. We use this framework with the caveat that it does not automatically validate behavioral mutation of Mealy machines, which is our domain.

**Automata learning.** Learning the reference model through queries is an alternative to manual modeling [3, 17], which we discuss as a complementary path.

**Contamination and generalization.** Evaluating LLMs on well-known domains risks rewarding memorization rather than reasoning, a recognized threat to the validity of such evaluations [23]. For this reason, we built fictional systems specifically for this study. They reduce direct item contamination, but they eliminate neither memorization nor the reuse of learned structural patterns; the argument they support is one of mitigation, not elimination.

# 4 Method and Experimental Design

## 4.1 Research Questions

- **RQ1 (quality of the inferred models).** To what extent do the inferred models satisfy formal validity, behavioral equivalence, and minimality, and how are they distributed across the resulting quality categories?
- **RQ2 (sufficiency of the metric).** Is the *reference-oracled input-sequence mutation score* sufficient as a proxy for the fidelity of the inferred models?
- **RQ3 (exploratory characterization of the inventory).** How do the results vary across systems and model configurations in the experimental inventory?

RQ2 is the central question of the study: a single non-equivalent model whose suite matches the reference suite's mutation score suffices for a negative answer. RQ3 is exploratory; it supports neither causal comparison nor a *ranking* of LLMs and is not part of the decisive evidence of the contribution.

## 4.2 Metric Evaluated in This Study

The metric evaluated here is a diagnostic measure defined for this study, motivated by mutation-based measures of test-suite effectiveness. It asks whether a downstream effectiveness signal—equality with the reference-derived mutation score—could substitute for direct behavioral checking when evaluating LLM-inferred models. We do not claim that this gap has been proposed in prior work as a fidelity criterion.

Let $\mathrm{bms}(S)$ be the fraction of mutants of the reference model killed by a suite $S$, over a fixed population of reference mutants. Let $S_{\mathrm{ref}}$ be the suite derived from the reference and $S_{\mathrm{inf}}$ the suite derived from the inferred model $M$. We call $\mathrm{bms}(S_{\mathrm{inf}})$ the *reference-oracled input-sequence mutation score* of $M$ and define the *gap* as

$$\mathrm{gap}(M) = \mathrm{bms}(S_{\mathrm{ref}}) - \mathrm{bms}(S_{\mathrm{inf}}). \tag{1}$$

The gap is computed over the same mutant population. For a suite derived from the inferred model, only its input sequences are used. Each sequence is executed on the reference and on each reference mutant, and a mutant is killed when its output trace diverges from the reference trace. The expected outputs prescribed by the inferred model therefore play no part in this decision.

Consequently, the metric measures the effectiveness of the selected sequences against the adopted mutant population rather than directly checking the inferred model's outputs. For $\mathrm{gap}(M) = 0$ to certify fidelity, it would be necessary that $\mathrm{gap}(M) = 0 \Rightarrow M \equiv R$. A single model satisfying $\mathrm{gap}(M) = 0$ and $M \not\equiv R$ is therefore sufficient to refute this implication.

## 4.3 Evaluation Process and Repair Cycle

The requirements make the system's input and output alphabets explicit. The first LLM response is converted into a structured machine and checked for completeness, determinism, and reachability. When the response is invalid, the next request includes the previous response and the list of errors produced by formal validation. These errors indicate, for example, missing transitions, nondeterminism, or unreachable states. The process continues until a valid model is obtained or the configured repair limit is reached. The use of an LLM to derive a state-machine representation from textual requirements follows the general setting explored in prior work [1, 15]. The structured output format, validation-feedback repair cycle, repair limit, and pipeline orchestration are experimental engineering choices of this study rather than proposed inference techniques.

We use *final attrition* to denote runs that remain invalid after this limit. They do not generate suites. For the remaining runs, equivalence, minimality, and test generation are evaluated on the *final model*; we therefore distinguish the validity of the initial attempt from the validity reached after any repairs.

For each valid model, we check equivalence with two complementary procedures: a synchronized traversal that compares outputs directly and a structural comparison after minimization. We then generate W, Wp, and HSI suites for equivalent and non-equivalent models alike. Equivalence is a fidelity verdict, not a generation filter. Each suite is executed against the same population of behavioral mutants of the reference, producing the *mutation score* and the *gap* defined in Equation 1. When the methods differ, we report W, Wp, and HSI separately; there is no aggregation across them.

Figure 1 summarizes the experimental process, making explicit the *feedback* repair cycle and the separation between the direct fidelity check and the evaluation of suite effectiveness.

## 4.4 Fidelity Classification

The classes are defined by two properties of the valid models: equivalence and minimality. They are mutually exclusive and, together with the invalid class, cover all pipeline outcomes. An equivalent and minimal model is classified as *faithful*; if it is equivalent but not minimal, it is *correct but redundant*; every valid, non-equivalent model is a *fidelity error*. Models that fail the formal check are *invalid*.

Minimality concerns behavioral redundancy rather than eligibility for test generation. A complete, deterministic, reachable machine can generate suites even if it contains behaviorally equivalent states. For this reason, valid non-minimal models are not discarded before test generation.

## 4.5 Equivalence Checking and Canonical Counterexamples

We use two procedures because they provide complementary checks of equivalence. The synchronized traversal compares the behaviors directly and produces a witness sequence. The structural approach

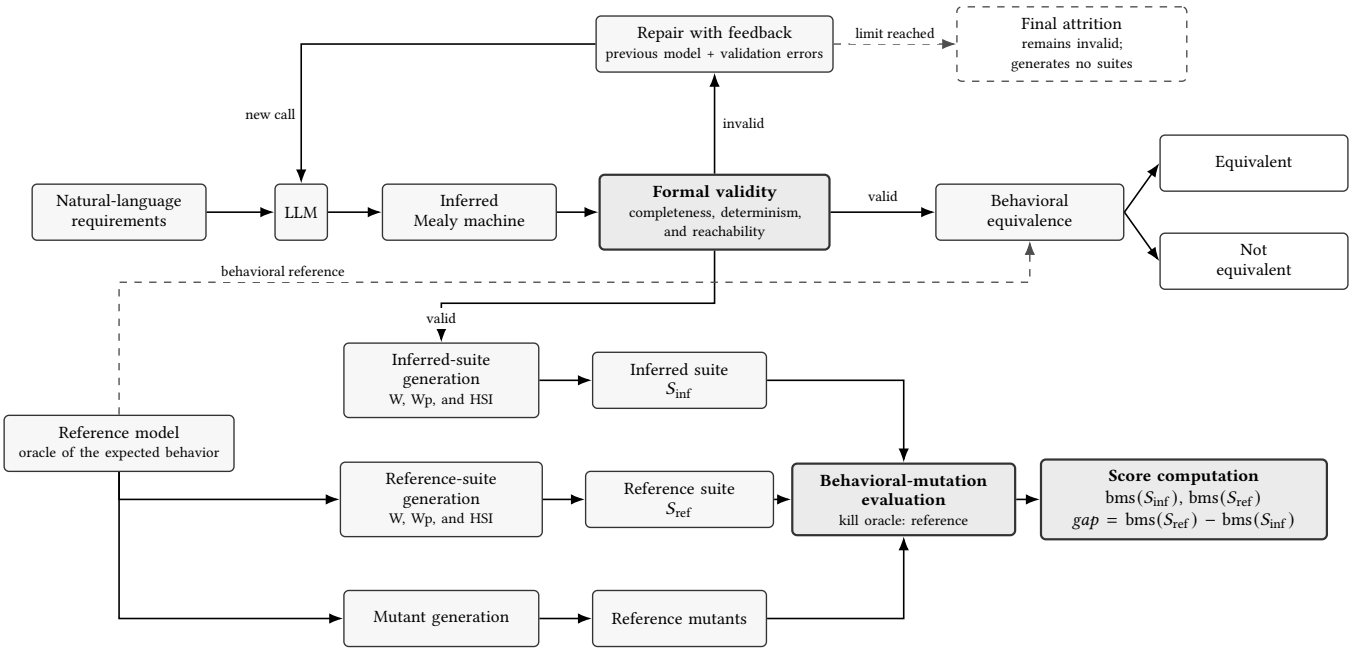


**Figure 1: Evaluation process for the inferred models. Initially invalid responses may go through repair with *feedback*; only runs that remain invalid after the limit are recorded as final attrition. Valid final models proceed, separately, to equivalence checking and to the evaluation of suite effectiveness.**

exploits the fact that two equivalent minimal machines are isomorphic [12]. Agreement between the procedures serves as an internal check, although they share components and do not constitute fully independent validations.

In the behavioral procedure, we traverse pairs of states from the initial states and, for each input, compare the outputs of the two machines. The first divergence demonstrates non-equivalence. In the structural procedure, we minimize the inferred machine by pruning unreachable states and refining the partition, and compare it with the minimal reference under isomorphism.

When several sequences expose the same non-equivalence, we choose a deterministic representation to ease reproduction and comparison across runs. We call the *canonical counterexample* the shortest sequence that reveals the first output divergence; ties are broken by the lexicographic order of the recorded alphabet.

Cross-check *with a Distinct Algorithm.* Since the two paths above share the output-comparison core, their agreement (set C) is expected by construction. For additional evidence, we implemented an alternative checker based on *exhaustive enumeration* of sequences: two complete, deterministic machines over the same alphabet are equivalent if and only if they agree on the output for every sequence of length up to $|Q_1| \cdot |Q_2|$ (product-automaton bound). This checker uses a distinct decision algorithm, although it shares the machine representation and the execution semantics. We compared the two on 305 pairs of small machines (five hand-built and 300 random, seed 42; 127 equivalent, 178 non-equivalent) and observed agreement on all 305 pairs. This increases confidence in the implementation but does not constitute a formal proof of correctness (scripts in the artifact package).

## 4.6 Repetition Protocol

In this paper, $N$ denotes the number of *pipeline runs*, not the number of individual API requests. The decisive cell (`gpt-5` × `tidewell`) was collected in two separate blocks, analyzed side by side and never combined into a single $N$.

**A1: retrospective analysis.** The first block gathered $N = 30$ runs under the same specification and configuration. Each run began with a request to the model; when the initial response was formally invalid, the pipeline could send a repair request with the previous response and the validation errors. In 28 runs, the first attempt already produced a valid model; R02 and R28 required one repair each, for a total of 32 API requests. No run was discarded or replaced. Final attrition was zero (0/30), and the two decisive counterexamples, R29 and R30, were valid on the first attempt.

**A2: prospective replication.** We repeated the same cell in $N = 50$ runs with the fixed *snapshot* `gpt-5-2025-08-07`. We kept the system, requirements, `zero_shot` mode, alphabets, *parsing*, formal validation, and repair policy unchanged. The data-collection protocol and the analysis pipeline were frozen before the prospective runs began. The 50 runs corresponded to 52 requests: two validity repairs (A2-R013 and A2-R029) and no transport retries. The responses and the required artifacts were preserved, and the analysis was performed locally, without new requests to the model.

A1 and A2 are separate collections over the same experimental cell. The remaining systems and configurations have fewer runs and are treated as exploratory; they are not incorporated into the decisive blocks.

### 4.7 Mutants and the Transition-Removal Operator

The behavioral mutants are first-order and are generated by exhaustive enumeration over four operators. The `transition_removal` operator removes one transition and produces a partial machine. Upon reaching an undefined transition, the executor halts the simulation and emits an undefined-output marker (`halt+None` semantics). These partial mutants are not covered by the completeness hypotheses of W, Wp, and HSI and are therefore excluded from any completeness claim. All generated mutants are included in the empirical denominator, including the partial mutants produced by `transition_removal`. The infrastructure does not perform dedicated equivalent-mutant detection, which we record as a limitation. In the central cases, the `tidewell_regulator` reference generates 2,411 mutants (output 1,680; target-state 660; transition-removal 60; initial-state 11) and `aurora_array` generates 3,149 (1,188; 1,836; 108; 17); the reported scores and *gaps* use these denominators. Section 5.4 recomputes the *gap* with and without the `transition_removal` operator to test whether the finding depends on partial mutants.

### 4.8 Target Systems

We adopted a purposive sample with two groups. The six canonical systems were selected to represent small MBT/FSM examples, such as *vending machine*, traffic light, authentication, *cruise control*, *infotainment*, and ATM, with two to seven states. They facilitate comparison, although their domains may have appeared in the LLMs' training data.

The three fictional systems were built to reduce direct contamination and to increase structural complexity while keeping the experimental cost feasible: `sentinel_latch` (12 states), `tidewell_regulator` (12), and `aurora_array` (18). Their requirements are functional and non-tabular. This selection is not meant to represent the population of state-based systems; the limitation is revisited under the external-validity threats.

### 4.9 Experimental Configurations

We varied the configuration of the inference model. The high-capacity model was `gpt-5` and the compact one `gpt-4o-mini`, both via the OpenAI API (*Chat Completions* endpoint); the descriptive inventory also includes `qwen2.5-coder` 3B and 7B, via Ollama, and `gemini-2.5-flash`. These configurations differ simultaneously in architecture, training data, scale, generation strategy, and provider. We therefore do not interpret observed differences as an isolated causal effect of capacity.

Each run starts with an independent request. Repair requests, when needed, include the previous response and the errors from formal validation. We set the temperature to 0.7 for gpt-4o-mini and gemini-2.5-flash. For gpt-5, the interface did not send this parameter, so the provider's default value was used. The output limit was `max_completion_tokens=16384`. We sent neither `reasoning.effort`, `top_p`, nor a sampling *seed*; *seed* 42 refers to mutant generation. The confidence intervals are descriptive and use the Wilson score method [21] instead of the Wald interval [5].

### 4.10 Traceability and Reproducibility

In the retrospective block A1, we persisted the *templates*, the requirements, the prompt identifiers, the textual responses, and the final models. The rendered prompts can be reconstructed from these artifacts and the code version, but they were not preserved. Also not recorded were the versioned *snapshot*, the response envelope, the response and request identifiers, the token counts, and the SDK version; the exact HTTP *payload* was not saved either.

In the prospective block A2, we fixed the *snapshot* `gpt-5-2025-08-07` and preserved the arguments passed to the SDK, the rendered prompts, the returned envelope, the identifiers, the *tokens*, and the attempt history, though not the byte-level HTTP body. This difference in traceability is accounted for in the threats to validity and in the artifact availability.

### 4.11 Experimental Sets

Table 1 summarizes A1, A2, B, C, and the exploratory pilot. A1 is contained in B, whereas A2 is a separate prospective collection. The pilot overlaps B in 21/24 runs; its remaining three are earlier high-capacity `tidewell` executions distinct from A1. Results are reported under their respective denominators and are never pooled.

## 5 Results

### 5.1 RQ1: Quality of the Inferred Models (Sets A1 and A2)

The results show that formal validity does not imply equivalence. Of the 30 runs (Table 2), 28 produced a formally valid machine on the first call to the model; R02 and R28 required one additional repair call. All 30 final models were valid and minimal, so there was no final attrition in block A1. Regarding fidelity, 28/30 final models were equivalent to the reference and classified as faithful (Wilson 95%: [0.79; 0.98]). This group includes R02 and R28, which became faithful after repair. R29 and R30, in turn, were fidelity errors: valid, minimal, yet non-equivalent (Wilson 95%: [0.02; 0.21]) and obtained on the first call, without repair. In the prospective block A2, 48/50 runs were valid on the first attempt, and two required one repair each (A2-R013 and A2-R029); all 50 final models were valid and minimal, 43/50 equivalent (faithful) and 7/50 fidelity errors (Wilson 95%: [0.07; 0.26]). The seven non-equivalent models were minimal and had the same size as the reference (12 states). Size and minimality, therefore, did not reveal the infidelity.

### 5.2 RQ2: Sufficiency of the Metric (Sets A1 and A2)

The A1 and A2 results show that the score is not sufficient as a proxy for fidelity. For the non-equivalent models R29 and R30, the difference in effectiveness was zero: the reference suite and the inferred suite killed all mutants ($\mathrm{bms}(S_{\mathrm{ref}}) = \mathrm{bms}(S_{\mathrm{inf}}) = 1.0$; all three methods—HSI, W, and Wp—yielded 1.0), resulting in $gap = 0$. The other 28 runs also had $gap = 0$ and corresponded to equivalent models. In this set, the metric did not distinguish the 28 faithful

**Table 1: Experimental sets. Results are reported under their own denominators and are never pooled; A1 is contained in B, whereas A2 is a separate prospective collection.**

| Set | Configuration / scope | Purpose |
|---|---|---|
| A1 (decisive) | high capacity × `tidewell` ($N$=30) | retrospective analysis (RQ2) |
| A2 (prospective) | fixed *snapshot* × `tidewell` ($N$=50) | prospective replication (RQ2) |
| B (descriptive) | 104 real pipeline runs | stratified inventory (RQ1/RQ3) |
| C (verification) | 187 valid models | implementation consistency |
| Pilot (explor.) | fictional systems, few runs | exploratory analysis (RQ3) |

**Table 2: Model quality per set. "Final valid" refers to each run's final model; A1 and A2 are reported separately. B is descriptive, and the pilot is exploratory. Wilson 95% intervals where applicable.**

| Set | Configuration / system | $N$ | Final valid | Equivalent | Minimal | Faithful | Status |
|---|---|---|---|---|---|---|---|
| A1 | high cap. × `tidewell` | 30 | 30/30 | 28/30 [0.79;0.98] | 30/30 | 28/30 | retrospective |
| A2 | fixed *snap.* × `tidewell` | 50 | 50/50 | 43/50 [0.74;0.93] | 50/50 | 43/50 | prospective |
| B | gpt-5 (all systems) | 36 | 36/36 | 34/36 | – | 33/36 | descriptive |
| B | gpt-4o-mini (all systems) | 19 | 5/19 | 2/19 | – | 2/19 | descriptive |
| B | real-run inventory (all models) | 104 | 84/104 | 78/104 | – | 77/104 | descriptive |
| Pil. | high cap. × 3 fictional | 9 | – | 8/9 | – | 7/9 | exploratory |
| Pil. | compact × 3 fictional | 15 | – | 0/15 | – | 0/15 | exploratory |

models from the 2 unfaithful ones. These two cases therefore refute $\text{gap}(M) = 0 \Rightarrow M \equiv R$. R29 and R30 were obtained on the first call and did not go through the repair cycle, so the decisive counterexample is not a product of the *feedback* correction step.

The reference suite and the R29/R30 suites reached the upper bound of the score (joint saturation, discussed below); the `aurora_array` case shows that the metric is not invariably saturated.

Figure 2 summarizes the two observed regimes. In both cases, the model is non-equivalent, but only in the visible case is there a loss of effectiveness against the mutant population.

**The invisible counterexample (R29 and R30).** Table 3 summarizes the contrasting cases. In R29 and R30, the reference has 12 states and the inferred model has 14 states, already minimal; this is a fidelity error, not redundancy. The canonical counterexample has length 7: `start, hi, start, lo, stop, start, start`. The divergence occurs at the seventh input (`start`), in state `RUN_OK`, where the reference produces `running` and the inferred model produces `run_low`. Because the mutants' kill oracle is the reference, the sequences of the inferred suite still kill all mutants; the inferred model's wrong output plays no part in this criterion. The *gap* therefore remains zero.

*The prospective replication (A2).* Block A2 reproduced the invisible regime under a protocol frozen before collection: 7 of the 50 runs produced non-equivalent models and, in *all* seven, W, Wp, and HSI killed the 2,411 mutants, with $gap = 0$ under all three methods. The seven models are behaviorally equivalent to one another and have 12 states, like the reference. They represent seven occurrences of the same error mode, not seven distinct error modes. The canonical counterexample has length 4: `start, hi, stop, start`; at the fourth input (`start`), the reference produces `priming` and the inferred model produces `running`. As in A1, the wrong output plays no part in the mutant-kill criterion (whose oracle is the reference), and the *gap* remains zero (Table 3).

**The visible counterexample (`aurora_array`).** In `aurora_array` (EXP-009-R02, compact model), the model is valid, minimal, and non-equivalent, with a counterexample of length 2 (`A, A`). On the second input, `A`, the reference produces `deny` and the inferred model produces `advance`. The change in the sequences selected by the suites reduced the inferred suite's ability to kill mutants. The three methods showed different values: the W score was 0.493 ($gap = 0.507$), while the HSI and Wp scores were 0.482 ($gap = 0.518$). All indicate a loss of effectiveness ($gap > 0$) and support the same qualitative interpretation; Table 3 presents the three values.

In the three representative cases, the error is an output error in valid, non-equivalent models. What differs is its effect on the sequences selected by the suite. In `aurora`, the change reduces the ability to kill mutants and makes the difference observable. In A1 and A2, that ability is preserved and the *gap* remains zero. Because the cases also differ in system, model configuration, and witness length, this contrast is illustrative rather than causal.

## 5.3 Explaining the Zero Gap: Suite-by-Suite Comparison

The zero *gap* of R29/R30 does not come from identical suites, but from different suites that kill exactly the same mutants. Table 4 recomputes, directly from the versioned machines, the sizes of the reference and inferred suites, the Jaccard index of their input sequences, and the difference between the sets of killed mutants (script `analyze_invisible_cases.py`). We distinguish three situations: (i) identical input suites; (ii) different suites that kill the same mutants; (iii) different suites with a loss of effectiveness.

In the invisible case (R29/R30), the suites are *not* identical: the Jaccard index ranges from 0.37 (W) to 0.70 (HSI), and the inferred

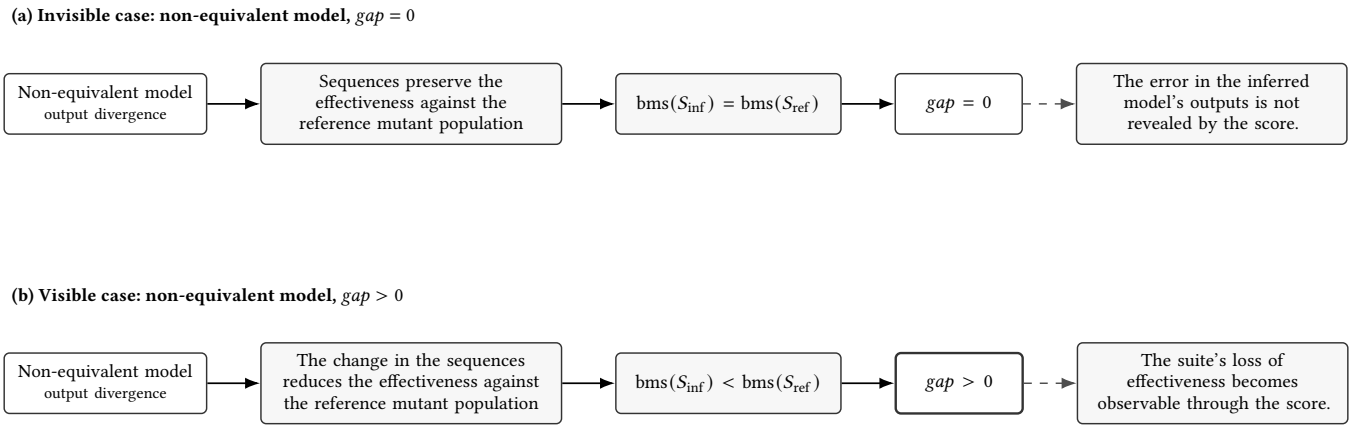


**Figure 2: Contrasting cases in the relation between fidelity and effectiveness. In the invisible case, the inferred suite retains the same score as the reference suite despite the non-equivalence. In the visible case, the loss of effectiveness makes the difference observable. Equivalence compares the models' outputs; the score compares the suites' effectiveness using the reference as the mutants' kill oracle.**

**Table 3: Contrasting cases, with the three generation methods (W, HSI, and Wp) reported explicitly. All models are valid, minimal, and non-equivalent, and in all of them $\text{bms}(S_{\text{ref}}) = 1.0$. The A2 row summarizes the seven non-equivalent prospective runs, which exhibit the same unfaithful behavior (a single error mode). In each case, the three methods support the same qualitative interpretation.**

| Case | ref/inf states | $\lvert w \rvert$ | Error (input: ref/inf) | Method | $\text{bms}_{\text{inf}}$ / *gap* | Regime |
|---|---|---|---|---|---|---|
| R29/R30 / tidewell | 12/14 | 7 | `start`: running/run_low | W | 1.0 / **0.0** | invisible |
| | | | | HSI | 1.0 / **0.0** | |
| | | | | Wp | 1.0 / **0.0** | |
| A2 / tidewell | 12/12 | 4 | `start`: priming/running | W | 1.0 / **0.0** | invis. (prosp.) |
| | | | | HSI | 1.0 / **0.0** | |
| | | | | Wp | 1.0 / **0.0** | |
| R02 / aurora | 18/11 | 2 | `A`: deny/advance | W | 0.493 / **0.507** | visible |
| | | | | HSI | 0.482 / **0.518** | |
| | | | | Wp | 0.482 / **0.518** | |

suite has more sequences than the reference one (for example, 270 versus 183 in W). Even so, both kill all 2,411 mutants *and* the sets of killed mutants are identical ("ref. only" = "inf. only" = 0), so $gap = 0$. The score is insensitive to the infidelity not because the sequences coincide, but because both suites saturate the mutant population. In the visible case (aurora), the suites also differ (Jaccard $\approx 0.35$–$0.39$), but now the inferred suite kills a strict subset of the mutants killed by the reference suite: 1,596 (W) or 1,632 (HSI/Wp) mutants are killed only by the reference and none only by the inferred suite, which produces $gap > 0$. The prospective block A2 exhibits the same pattern as the invisible case: for the seven unfaithful models, the inferred and reference suites have nearly equal sizes (W 183/183; Wp 109/110; HSI 91/92) and, although they differ (W Jaccard $\approx 0.50$), they kill exactly the same 2,411 mutants, with $gap = 0$.

## 5.4 Sensitivity of the Gap to Mutation Operators

Because both suites kill all 2,411 `tidewell` mutants, zero *gap* necessarily persists for any subset of this population. We nevertheless report the operator breakdown, recomputed from the recorded per-operator reports (script `operator_sensitivity.py`), to document that the decisive counterexamples do not rely on the inclusion of partial mutants. Table 5 also examines the non-saturated `aurora` case, where removing `transition_removal` still leaves the *gap* positive.

Excluding `transition_removal` reduces the `tidewell` population from 2,411 to 2,351 mutants and the `aurora` population from 3,149 to 3,041 (Table 5). For R29/R30, the *gap* is also zero within each operator class (detailed results in `operator_sensitivity.csv`). In `aurora`, it remains positive (0.509 in W; 0.520 in HSI/Wp). Thus, excluding partial mutants does not change either regime, although

**Table 4: Suite-by-suite comparison, recomputed from the versioned machines. $|S_{\text{ref}}|$ and $|S_{\text{inf}}|$ are numbers of distinct input sequences; "ref. only" and "inf. only" are mutants killed exclusively by one of the suites. R29 and R30 produce the same values (same defect).**

| Case | Method | $\lvert S_{\text{ref}}\rvert$ | $\lvert S_{\text{inf}}\rvert$ | Jaccard | ref. only | inf. only | *gap* |
|---|---|---|---|---|---|---|---|
| R29/R30 / tidewell | W | 183 | 270 | 0.369 | 0 | 0 | 0.0 |
| | Wp | 109 | 133 | 0.603 | 0 | 0 | 0.0 |
| | HSI | 91 | 111 | 0.698 | 0 | 0 | 0.0 |
| R02 / aurora | W | 436 | 268 | 0.386 | 1596 | 0 | 0.507 |
| | Wp | 218 | 172 | 0.388 | 1632 | 0 | 0.518 |
| | HSI | 192 | 152 | 0.349 | 1632 | 0 | 0.518 |

it does not eliminate the ceiling effect. The seven prospective A2 cases likewise retain zero *gap* in each operator class.

## 5.5 Inventory, Consistency, and Pilot (B, C, and RQ3)

The descriptive blocks play no part in the prospective replication and are reported compactly. In inventory B (104 real pipeline runs; not a quality rate), 77 models were faithful, 1 correct but redundant, 6 fidelity errors, and 20 invalid, with the invalid ones concentrated in two configurations (14 and 6) and none in the high-capacity one. In set C, of the 207 models associated with a reference (103 reference-mirror positive controls and 104 real runs), 187 are valid and the structural and behavioral verdicts agree in all 187 valid cases. This is an implementation-consistency check, expected by construction and complemented by the exhaustive-enumeration *cross-check* (Section 4.5); it does not constitute a formal proof. As an exploratory analysis (RQ3), on the three fictional systems, the high-capacity configuration produced 7/9 faithful models and 8/9 equivalent models, whereas the compact configuration produced 0/15 faithful models; these results come from small, confounded samples, do not constitute a *ranking* of LLMs, and enter neither the contributions nor the decisive evidence.

## 6 Discussion

**Answer to RQ1: formal validity does not certify fidelity.** In A1, the 30/30 valid final models include 2 non-equivalent ones; in A2, the 7 unfaithful models are valid, minimal, and the same size as the reference. Formal validity enables suite generation, but it does not certify that the behavior is correct.

**Answer to RQ2: the score is not sufficient as a proxy for fidelity.** Block A2, under a fixed *snapshot* and a protocol frozen before collection, reproduced the invisible regime: 7/50 non-equivalent models with $gap = 0$ in W, Wp, and HSI. A1 and A2 produced different error modes: in A1, `running`/`run_low`, with a counterexample of length 7 and 12/14 states; in A2, `priming`/`running`, with a counterexample of length 4 and 12/12 states. The two blocks reproduce the same metric-level phenomenon, but not the same defect. We performed no inferential comparison between the blocks' rates; the intervals are descriptive. The 7/50 result does not represent a general failure rate of `gpt-5`, and A2 does not estimate the prevalence of the phenomenon.

**When the score is informative.** Since the kill oracle is the reference, the metric measures the effectiveness of the suite's sequences, not the correctness of the outputs prescribed by the inferred model. The score therefore answers a different question: how effective the sequences are against the mutant population, not whether the inferred model is equivalent to the reference. In R29/R30, this yields $gap = 0$ despite the non-equivalence, whereas in `aurora` the loss of effectiveness makes the difference observable ($gap > 0$).

**Ceiling effect and the conceptual limitation.** On the `tidewell` mutant population, both the reference suite and the R29/R30 suites reached the upper bound of the score. This *joint saturation* prevents the metric from distinguishing them on that population, even though their sequences differ. Note that $\text{bms}(S_{\text{ref}}) = 1$ does not force $\text{bms}(S_{\text{inf}}) = 1$: as in aurora, the inferred suite could score lower and produce $gap > 0$; what happened in R29/R30 is that both suites, although different, killed the entire mutant population. The sensitivity analysis (Section 5.4) shows that the result does not depend exclusively on the `transition_removal` operator, but it does *not* eliminate the ceiling effect in the remaining mutant subpopulations. Beyond the ceiling effect, the conceptual limitation remains: the score does not directly check the outputs predicted by the inferred model, because the reference remains the kill oracle. An error can become visible when it changes the generated sequences, as in aurora; but this occasional visibility does not turn $gap = 0$ into a certificate of equivalence.

**A two-layer evaluation.** The results suggest evaluating the inference first by formal eligibility and behavioral equivalence, and then by the effectiveness of the generated suites. The second layer complements the first, but it cannot replace it. When no reference model is available, the *gap* defined in this study cannot be computed. Structure-oriented criteria and execution-based output assessment have been used to evaluate LLM-generated state machines [1, 15, 22], while mutation-based measures can assess the effectiveness of generated tests [9, 14]. Such measures may help identify weak or suspicious results, but they assess observable properties or test effectiveness rather than certify behavioral fidelity with respect to an unavailable reference. A positive *gap* indicates reduced sequence effectiveness against the adopted mutant population; it is a diagnostic signal, not a fidelity verdict.

**Answer to RQ3 and relation to the literature.** The inventory suggests variation across systems and configurations, but the small, confounded samples support neither a ranking of LLMs nor causal

**Table 5: Sensitivity of the *gap* to the mutation operators. Each row is a subpopulation with its own denominator $n$; the *gaps* check the persistence of the regime and are not comparable across rows. "Without `transition_removal`" coincides with the operators that preserve completeness.**

| Case | Mutant population | W *gap* | Wp *gap* | HSI *gap* | $n$ |
|---|---|---|---|---|---|
| R29/R30 / tidewell | all operators | 0.0 | 0.0 | 0.0 | 2411 |
| | without `transition_removal` | 0.0 | 0.0 | 0.0 | 2351 |
| R02 / aurora | all operators | 0.507 | 0.518 | 0.518 | 3149 |
| | without `transition_removal` | 0.509 | 0.520 | 0.520 | 3041 |

comparison. Prior work evaluates LLM-inferred FSMs by structure, output matching, or use in test generation [1, 15, 22], while other work employs mutation to guide or evaluate LLM-generated tests [9, 14]. Our result connects these threads: the effectiveness of the sequences, even when measured against the reference as the oracle, does not replace direct verification of the intermediate model. Evaluating only the suite can therefore leave fidelity errors undetected.

## 7 Threats to Validity

*Construct Validity.* Behavioral mutation is not the same as code-level *mutation testing*, and mutants are not real faults; the mutant population may not represent all relevant defects. We do not perform a dedicated equivalent-mutant analysis. In general, equivalent mutants would affect the denominator and therefore the interpretation of absolute mutation scores. In the decisive `tidewell` cases, however, every generated mutant was killed by both suites; under the adopted execution semantics, a behaviorally equivalent mutant could not be killed. Equivalent mutants therefore do not explain the observed zero-gap counterexamples. The *mutation score* measures the effectiveness of the sequences against this population and does not directly check the inferred model's outputs; joint saturation of the suites can prevent the distinction even when they differ. The *gap* depends on the population and is not invariant to the choice of operators; it requires a reference model, since the mutants, the suite, and the kill oracle derive from it. The `transition_removal` operator produces partial machines, outside the completeness hypotheses of W, Wp, and HSI. The reference model used as the oracle is itself human-constructed.

**Internal validity.** The differences between model configurations confound capacity with architecture, data, scale, generation strategy, and provider; there is no causal design. In A1, R02 and R28 went through the repair cycle; in A2, A2-R013 and A2-R029 also required one additional request. The final models of both blocks thus combine responses that were valid on the first attempt and repaired responses. We treat repair as part of the evaluated pipeline. The decisive non-equivalent models in A1 and A2, however, were obtained on the first attempt, so the phenomenon was not an artifact of feedback-based repair. Different numbers of runs across systems prevent direct comparisons of raw counts.

**External validity.** There are few independent systems (six remodeled canonical ones and three fictional ones). The decisive evidence comes from the `gpt-5` × `tidewell` cell; repetitions on the same system are not independent systems (pseudoreplication), and the Wilson intervals are used descriptively. The fictional systems mitigate direct item contamination but do not eliminate memorization. The prospective block A2 mitigates retrospective selection and uses a fixed *snapshot*, but it still covers only one model *snapshot*; its seven unfaithful models exhibited the same behavior and represent repeated occurrences of one error mode, not seven independent types of infidelity. Nor do the 50 runs correspond to 50 independent systems. There is no prevalence estimate; generalization to other systems, *prompts*, LLMs, and operators requires new studies.

**Conclusion validity.** The insufficiency claim is established by counterexample: a single non-equivalent model with $gap = 0$ suffices, and the study provides such cases for the adopted *gap* definition and mutant population. There is a risk of implementation error; some checks share components, so the consistency of set C is expected by construction. We mitigate part of this risk with the exhaustive-enumeration *cross-check* (Section 4.5), which agreed with the main checker on 305/305 pairs, but which is also implementation evidence, not formal proof.

**Reproducibility.** In the retrospective block A1, the runs against proprietary APIs were recorded by the model *alias*, without the versioned *snapshot*. The response envelope was discarded at the source, so `response.model`, request identifiers, *usage*, and token counts were not retained in the artifacts; the SDK version was not frozen and the exact *payload* was not saved. The rendered prompts are reconstructible (templates, requirements, and code version), but they were not originally persisted. In contrast, the textual content of the responses, the final models, the scripts, and the additional analyses are persisted and re-executable over the versioned artifacts, without new API calls. These limitations affect the retrospective block A1; the prospective block A2 froze the protocol and the analysis *pipeline* before collection, fixed the *snapshot* `gpt-5-2025-08-07`, and preserved the arguments sent to the SDK, the response envelopes, the identifiers, and the *tokens*, reducing the threats of an unknown *snapshot* and missing metadata.

## 8 Conclusion

We studied how to evaluate LLM-inferred Mealy machines for model-based testing. The results support three main conclusions:

(1) Formal validity, minimality, and size do not guarantee behavioral equivalence: in A1, the 30 final models were valid and minimal, but two were not equivalent; in A2, the seven unfaithful models were valid, minimal, and the same size as the reference.

(2) The retrospective finding was replicated prospectively: in block A2, under a frozen protocol, seven runs produced non-equivalent models with $gap = 0$ in W, Wp, and HSI. These seven models share the same error mode, which differs from the one observed in A1. This zero *gap* does not stem from identical suites and persists even when the partial mutants are excluded. In the `aurora_array` case, the change in the selected sequences produced $gap > 0$, showing that the metric can reveal losses in effectiveness. Nevertheless, $gap = 0$ does not certify fidelity.
(3) The evaluation of inferred models should separate three properties: formal validity, behavioral equivalence, and suite effectiveness. Fidelity should be the primary criterion, and the *reference-oracled input-sequence mutation score* a complementary measure.

A1 and A2 are separate collections of the same cell and do not estimate the general prevalence of the phenomenon. We also make no claims about the superiority of any LLM or the inadequacy of W, Wp, or HSI. The contribution is to show that sequence effectiveness cannot replace checking the fidelity of the model that generated those sequences.

## Artifact Availability

The artifact package contains the data, models, prompts, scripts, and results needed to inspect and re-execute the reported analyses offline. It is publicly available at https://github.com/alecsmatos1/fidelity-is-not-effectiveness-artifact.

## Acknowledgments

The authors acknowledge the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), through the Programa de Excelência Acadêmica (PROEX). The authors also acknowledge the support of the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under processes 306719/2025-8 and 406941/2025-4. The authors thank Adelson Carlos Madruga for his careful reading of the manuscript and valuable comments.

## Declaration on the Use of Generative AI

ChatGPT (OpenAI) and Grammarly were used only for language revision and improvement of clarity. All scientific content, analyses, and conclusions were produced and verified by the authors.

## References

[1] Samer Abdulkarim, Evan Boyd, Karl Bridi, Alec Tufenkjian, Boqi Chen, and Gunter Mussbacher. 2026. Structure- and Event-Driven Frameworks for State Machine Modeling with Large Language Models. arXiv:2604.00275 [cs.SE] doi:10.48550/arXiv.2604.00275
[2] James H. Andrews, Lionel C. Briand, and Yvan Labiche. 2005. Is Mutation an Appropriate Tool for Testing Experiments?. In *Proceedings of the 27th International Conference on Software Engineering (ICSE)*. Association for Computing Machinery, New York, NY, USA, 402–411. doi:10.1145/1062455.1062530
[3] Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. *Information and Computation* 75, 2 (1987), 87–106. doi:10.1016/0890-5401(87)90052-6
[4] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The Oracle Problem in Software Testing: A Survey. *IEEE Transactions on Software Engineering* 41, 5 (2015), 507–525. doi:10.1109/TSE.2014.2372785
[5] Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. 2001. Interval Estimation for a Binomial Proportion. *Statist. Sci.* 16, 2 (2001), 101–133. doi:10.1214/ss/1009213286
[6] Tsun S. Chow. 1978. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering* SE-4, 3 (1978), 178–187. doi:10.1109/TSE.1978.231496
[7] Rita Dorofeeva, Khaled El-Fakih, Stéphane Maag, Ana R. Cavalli, and Nina Yevtushenko. 2010. FSM-based Conformance Testing Methods: A Survey Annotated with Experimental Evaluation. *Information and Software Technology* 52, 12 (2010), 1286–1297. doi:10.1016/j.infsof.2010.07.001
[8] Sandra C. P. F. Fabbri, Márcio Eduardo Delamaro, José Carlos Maldonado, and Paulo César Masiero. 1994. Mutation Analysis Testing for Finite State Machines. In *Proceedings of the 5th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE Computer Society Press, Los Alamitos, CA, USA, 220–229. doi:10.1109/ISSRE.1994.341378
[9] Christopher Foster, Abhishek Gulati, Mark Harman, Inna Harper, Ke Mao, Jillian Ritchey, Hervé Robert, and Shubho Sengupta. 2025. Mutation-Guided LLM-Based Test Generation at Meta. In *Companion Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion)*. Association for Computing Machinery, New York, NY, USA, 180–191. doi:10.1145/3696630.3728544
[10] Susumu Fujiwara, Gregor von Bochmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. 1991. Test Selection Based on Finite State Models. *IEEE Transactions on Software Engineering* 17, 6 (1991), 591–603. doi:10.1109/32.87284
[11] René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. 2014. Are Mutants a Valid Substitute for Real Faults in Software Testing?. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE)*. ACM, New York, NY, USA, 654–665. doi:10.1145/2635868.2635929
[12] David Lee and Mihalis Yannakakis. 1996. Principles and Methods of Testing Finite State Machines—A Survey. *Proc. IEEE* 84, 8 (1996), 1090–1123. doi:10.1109/5.533956
[13] Gang Luo, Alexandre Petrenko, and Gregor von Bochmann. 1995. Selecting Test Sequences for Partially-Specified Nondeterministic Finite State Machines. In *Protocol Test Systems VII*. Springer, Boston, MA, USA, 95–110. doi:10.1007/978-0-387-34883-4_6
[14] Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. 2024. Effective Test Generation Using Pre-Trained Large Language Models and Mutation Testing. *Information and Software Technology* 171 (2024), 107468. doi:10.1016/j.infsof.2024.107468
[15] Omer Nguena Timo, Paul-Alexis Rodriguez, and Florent Avellaneda. 2026. Designing FSMs Specifications from Requirements with GPT 4.0. arXiv:2603.29140 [cs.SE] doi:10.48550/arXiv.2603.29140
[16] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A Taxonomy of Model-Based Testing Approaches. *Software Testing, Verification and Reliability* 22, 5 (2012), 297–312. doi:10.1002/stvr.456
[17] Frits Vaandrager. 2017. Model Learning. *Commun. ACM* 60, 2 (2017), 86–95. doi:10.1145/2967606
[18] Frits Vaandrager and Ivo Melse. 2025. New Fault Domains for Conformance Testing of Finite State Machines. In *36th International Conference on Concurrency Theory (CONCUR 2025) (LIPIcs, Vol. 348)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 34:1–34:22. doi:10.4230/LIPIcs.CONCUR.2025.34
[19] M. P. Vasilevskii. 1973. Failure Diagnosis of Automata. *Cybernetics* 9, 4 (1973), 653–665. doi:10.1007/BF01068590
[20] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, and Qing Wang. 2024. Software Testing With Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering* 50, 4 (2024), 911–936. doi:10.1109/TSE.2024.3368208
[21] Edwin B. Wilson. 1927. Probable Inference, the Law of Succession, and Statistical Inference. *J. Amer. Statist. Assoc.* 22, 158 (1927), 209–212. doi:10.1080/01621459.1927.10502953
[22] Yuheng Wu, Berk Gokmen, Zhouhua Xie, Peijing Li, Caroline Trippel, Priyanka Raina, and Thierry Tambe. 2026. LLM-FSM: Scaling Large Language Models for Finite-State Reasoning in RTL Code Generation. arXiv:2602.07032 [cs.AR] doi:10.48550/arXiv.2602.07032
[23] Cheng Xu, Shuhao Guan, Derek Greene, and M-Tahar Kechadi. 2024. Benchmark Data Contamination of Large Language Models: A Survey. arXiv:2406.04244 [cs.CL] doi:10.48550/arXiv.2406.04244