AUTARK Academic V3

Testmethodik und Anti-Goodhart-Prinzipien

Stand der dokumentierten Evidenz: 30. August 2026

---

1. Ziel der Testmethodik

AUTARK Academic V3 wird nicht nur danach beurteilt, ob einzelne Antworten beeindruckend wirken. Entscheidend ist, ob sich belastbar zeigen lässt,

- welche Aufgaben das System tatsächlich löst,
- wie zuverlässig es sie technisch bis zu einer verwertbaren Antwort ausführt,
- wie oft die fachliche Antwort korrekt ist,
- welche Teile der Architektur zum Ergebnis beigetragen haben,
- und ob Verbesserungen auch auf **neuen, zuvor nicht zur Optimierung verwendeten Aufgaben** bestehen bleiben.

Das Kernproblem bei der Evaluation eines selbst entwickelten KI-Systems ist **Goodharting**:

> *Sobald ein Messwert zum direkten Optimierungsziel wird, steigt die Gefahr, dass das System auf genau diesen Messwert hin verbessert wird, ohne dass die zugrunde liegende allgemeine Leistung im gleichen Maß wächst.*

Bei LLM-Systemen kann das sehr subtil geschehen. Schon kleine Entscheidungen können einen Benchmark nachträglich „freundlicher“ machen:

- dieselben Aufgaben wiederholt testen;
- nach Kenntnis der Fehler gezielt Prompts ändern;
- schlechte Fälle aus der Statistik entfernen;
- technische Ausfälle als „fast richtig“ behandeln;
- bei einem unerwarteten Resultat den Antwortschlüssel umdeuten;
- nur erfolgreiche Modellrouten berichten;
- einen ausgewählten Failure-Recovery-Test wie einen neuen allgemeinen Benchmark interpretieren;
- verschiedene Entwicklungsstufen statistisch zusammenwerfen;
- oder nachträglich ein Testdesign ändern, bis das gewünschte Ergebnis entsteht.

Die AUTARK-V3-Methodik versucht deshalb nicht, Goodharting „psychologisch zu vermeiden“, sondern baut **prozedurale Barrieren** dagegen.

Dabei ist auch die **Zähleinheit** selbst Teil der Methodik. „48 Aufgaben“, „144 Task-Modell-Slots“, „288 Author-Runs“ und „mehrere hundert Graderbewertungen“ bezeichnen verschiedene Dinge und dürfen nicht zu einer einzigen scheinbar präzisen Benchmarkzahl zusammengeschoben werden.

---

2. Grundprinzip: Entwicklung und Evaluation werden getrennt

Ein zentraler Unterschied besteht zwischen:

Entwicklungsarbeit

Hier dürfen Fehler untersucht, Ursachen diagnostiziert, Prompts verbessert, Routingregeln verändert oder technische Limits angepasst werden.

Beispiele:

- ein technischer R671-Ausfall wird analysiert;

- ein Tokenlimit wird erhöht;
- ein Guard-Race wird repariert;
- ein Toolpfad wird neu gebunden;
- eine neue Routinghypothese wird entworfen.

Diese Arbeiten erzeugen wertvolle **Entwicklungsevidenz**, aber nicht automatisch einen neuen allgemeinen Leistungsnachweis.

Evaluation

Hier werden die Bedingungen vor dem Test festgelegt und anschließend nicht aufgrund des Ergebnisses verändert.

Dazu gehören:

- frische Aufgaben;
- vorab eingefrorene Testmengen;
- eingefrorene Reihenfolge;
- eingefrorene Antwortschlüssel;
- eingefrorene Gradingregeln;
- festgelegte Modelle, Seeds und Promptsemantik;
- vorher definierte Kriterien für technische Ausfälle;
- vorher definierte Regeln für Adjudikation;
- und ein klarer Abschlussstatus.

Diese Trennung ist eine der wichtigsten Anti-Goodhart-Barrieren.

---

3. Freeze before exposure

Bevor ein echter Evaluationslauf beginnt, werden zentrale Bestandteile des Tests **eingefroren**.

Typischerweise umfasst ein Freeze:

- Taskset
- Reihenfolge
- Antwortschlüssel
- Rubriken
- Modellzuweisungen beziehungsweise Routinglogik
- Grader
- Seeds
- Retryregeln
- technische Failure-Kriterien
- statistische Auswertung
- gegebenenfalls Selektions- oder Freigabeschwellen

Die Idee ist einfach:

> *Was nach Kenntnis der Antworten geändert wird, darf nicht mehr so behandelt werden, als sei es vor der Messung festgelegt worden.*

Ein Freeze wird deshalb als eigenes Artefakt gespeichert und mit SHA-256 gebunden.

---

4. Frische Holdouts statt wiederholter Optimierung auf denselben Aufgaben

Ein Ergebnis wird umso weniger aussagekräftig, je häufiger dieselben Aufgaben während der Entwicklung betrachtet wurden.

Deshalb verwendet AUTARK für entscheidende Bewertungen **fresh holdouts**:

- neue Aufgaben,
- neue Varianten,
- family-wise getrennte Aufgaben,
- oder prospektiv erzeugte Testsets.

Ein alter Test kann weiterhin zur Regression oder Diagnose dienen. Er gilt dann aber nicht mehr als unabhängiger Leistungsnachweis.

Der finale 48-Aufgaben-Test wurde deshalb nicht einfach aus einem alten Benchmark wiederholt, sondern als neuer, eingefrorener Holdout aufgebaut. **Er war die abschließende unabhängige Production-Prüfung, nicht der erste oder einzige Test des Systems.** Vor ihm lag bereits eine umfangreiche Entwicklungs- und Diagnosekette aus First Pass und Extensions B/C/A.

---

5. Umfang der empirischen Evidenz: 48 ist nicht die Gesamtzahl der Tests

Die auf der Webseite hervorgehobenen **48 neuen Aufgaben** gehören zum letzten frischen End-to-End-Holdout. Sie sind gerade deshalb wissenschaftlich wichtig, weil die früheren Aufgaben nach umfangreicher Analyse nicht mehr als unabhängige Bestätigung derselben Architektur dienen konnten.

Vor diesem finalen Holdout gab es jedoch bereits eine wesentlich größere experimentelle Kette. Für deren Umfang müssen verschiedene Zähleinheiten auseinandergehalten werden.

5.1 Author-seitige Evaluations-Slots vor dem finalen Holdout

Experiment	Design	Neue author-seitige Evaluations-Slots / Runs
First Pass	48 akademische Aufgaben x 3 lokale Modelle	**144**
Extension B	dieselben 48 Aufgaben durch das reale Operational System S	**48**
Extension C	24 FULL-Runs; auf einem diagnostischen 8er-Subset zusätzlich 8 NO_RETRIEVAL + 8 NO_FORMAL_TOOLS	**40**
Extension A	28 ausgewählte Task-Modell-Einheiten; je eine R1- und R2-Selbstrevision	**56**
Summe vor dem finalen frischen Holdout		**288**

Die Zahl **288** ist daher ein sinnvoller konservativer Zähler für die geplanten bzw. wissenschaftlich ausgewerteten **author-seitigen Task-Runs/Slots** in First Pass + B + C + A. Sie ist aber **kein Benchmark-N von 288 unabhängigen Aufgaben**. Viele Runs verwenden dieselben Grundaufgaben unter verschiedenen Modellen, Routen oder Interventionen.

Die korrekte Aussage lautet deshalb nicht:

> *„AUTARK wurde vor dem finalen Test an 288 unabhängigen Aufgaben getestet.“*

sondern etwa:

> ***„Vor dem finalen frischen 48-Aufgaben-Holdout lagen bereits 288 kontrollierte author-seitige Evaluations-Slots bzw. Runs aus Baseline-, Routing-, Tool-/Retrieval- und Selbstkorrekturexperimenten vor.“***

5.2 First Pass allein war bereits deutlich größer als „48 Tests“

Der First Pass bestand aus **48 Aufgaben x drei Modellen = 144 Task-Modell-Slots**. Die drei eingefrorenen Ergebnisvektoren waren:

- R1-671B: 45 C / 1 I / 1 U / 1 T
- Qwen3-235B: 42 C / 2 I / 4 U / 0 T
- DSV4: 40 C / 5 I / 1 U / 2 T

Schon diese 144 Slots führten zu **141 auswertbaren Antworten**. Jede dieser Antworten wurde den beiden Primärgradern J1 und J2 zugeführt; **89 Fälle** wurden nach den eingefrorenen Disagreement-Regeln zusätzlich für GLM-Adjudikation markiert, davon erhielten 85 ein gültiges GLM-Urteil. Die Grading-Arbeit liegt also bereits im First Pass im Bereich mehrerer hundert Bewertungseinheiten - zusätzlich zu den Solver-Slots.

5.3 Physische LLM-Requests sind nochmals zahlreicher

Ein wissenschaftlicher Task-Slot ist nicht zwingend identisch mit genau einem HTTP-/LLM-Request. Technische Retryregeln können mehrere physische Requests für denselben eingefrorenen Slot zulassen, ohne daraus mehrere akademische Datenpunkte zu machen.

Ein konkretes Beispiel: Für DSV4 wurden im First Pass 48 Task-Slots ausgewertet, aber die dokumentierte Versuchszahl war:

- 40 Slots mit 1 Versuch,
- 5 Slots mit 2 Versuchen,
- 1 Slot mit 3 Versuchen,
- 2 Slots mit 4 Versuchen.

Das entspricht **61 physischen DSV4-Author-Requests für 48 wissenschaftliche DSV4-Slots**. Schon deshalb wäre eine einzelne Gesamtzahl „LLM-Calls“ ohne Definition missverständlich.

5.4 Nach dem finalen Holdout ging die Evaluation weiter

Der finale 48er-Holdout war ebenfalls nicht das Ende der empirischen Arbeit. Danach folgten unter anderem:

- **RM16:** 24 frische Mathematik-/Physikaufgaben x 2 gematchte Author-Lanes (R70 und R671) = **48 weitere author-seitige Evaluationsruns**;
- **RM17.1.1A:** bislang **2** ausgewählte Failure-Recovery-Runs unter höherem Completion-Budget.

Allein für die hier explizit aufgezählten Stufen ergibt sich damit bis RM17.1.1A ein konservativer Zähler von

288 + 48 finaler Holdout + 48 RM16 + 2 RM17 = 386 author-seitigen Evaluations-Slots/Runs.

Diese **386** sind weiterhin **keine 386 unabhängigen Aufgaben** und auch **nicht die Gesamtzahl aller physischen Modellinferenzen** des Projekts. Nicht enthalten sind beispielsweise:

- Primärgrader- und GLM-Inferenzen der Extensions und späteren Tests,
- technische Retries,
- RM15-Kalibrations- und Revalidierungsläufe,
- Safety-/Lifecycle-/Fault-Injection-Canaries,
- Planner-, Tool- und Lean-Reattestationen,
- technische Preflights ohne akademische Exposition.

Die tatsächliche Zahl lokaler Modellinferenzen ist daher erheblich höher. Eine exakte globale Call-Zahl wäre nur nach einem gesonderten, artefaktübergreifenden Audit sinnvoll.

5.5 Warum diese Unterscheidung selbst Anti-Goodhart ist

Es wäre verlockend, die größte verfügbare Zahl zu nennen und daraus eine beeindruckende Benchmarkgroesse zu machen. Genau das wäre methodisch falsch.

AUTARK trennt deshalb mindestens:

1. **einzigartige Aufgaben**,
2. **Task-Modell-/Task-System-Evaluations-Slots**,
3. **Author-/Solver-Generationen**,
4. **physische Requests inklusive technischer Retries**,
5. **Grader-/Adjudikations-Inferenzen**,
6. **frische unabhängige Holdout-Aufgaben**.

Der finale Wert **44/48 = 91,67 %** bezieht sich ausschließlich auf den frischen finalen 48-Aufgaben-Holdout. Die früheren 288 author-seitigen Runs bilden die Entwicklungs- und Mechanismusevidenz, werden aber **nicht** mit dem Holdout zu einem nachträglichen Gesamtscore verrechnet.

Diese Trennung verhindert eine weitere Goodhart-Variante: Testvolumen wird transparent berichtet, ohne wiederverwendete Aufgaben oder diagnostische Interventionen als zusätzliche unabhängige Benchmarkdaten auszugeben.

---

6. Family-wise Design gegen subtile Leakage-Effekte

Ein Problem bei LLM-Benchmarks ist, dass zwei Aufgaben oberflächlich verschieden aussehen können, aber dieselbe mathematische, physikalische oder argumentative Struktur besitzen.

AUTARK verwendet deshalb **Taskfamilien**.

Im finalen 48er-Test:

- 24 Familien
- je zwei Varianten
- insgesamt 48 Aufgaben

Statistische Unsicherheit wird dabei nicht einfach so behandelt, als seien alle 48 Antworten unabhängig.

Stattdessen wurde ein **familiengeclusterter Bootstrap** verwendet.

Das ist konservativer als ein naiver Bootstrap über einzelne Aufgaben, weil zwei Varianten derselben Familie gemeinsame Fehlerursachen besitzen können.

---

7. Blindbewertung

Die wissenschaftliche Bewertung soll möglichst wenig davon wissen, welches System oder welche Route eine Antwort erzeugt hat.

Dafür werden Antworten in einen **Blind-Corpus** überführt.

Die Grader erhalten:

- die Aufgabe,
- die Antwort,
- die Bewertungsrubrik,

aber nicht die Information, welches Author-Modell die Antwort erzeugt hat, sofern diese Information für die Bewertung nicht nötig ist.

Damit wird reduziert, dass ein Grader unbewusst denkt:

> *„Das ist vom großen 671B-Modell, also wird es wohl richtig sein.“*

oder:

> *„Das ist nur vom 35B-Modell, also bin ich skeptischer.“*

Blindheit ist kein perfekter Schutz, aber eine wichtige Barriere gegen Erwartungseffekte.

---

8. Mehrstufiges Grading statt Einzelurteil

AUTARK verwendet nicht einfach einen einzelnen LLM-Grader als „Wahrheitsmaschine“.

Die typische Struktur ist:

- J1: primärer Grader
- J2: zweiter Bewertungsweg
- GLM: selektive Adjudikation bei Konflikten oder technischen Problemen

Die Regeln dafür werden vorher festgelegt.

Beispiel aus RM16:

- beide Primärgrader gültig und einig → gemeinsame Entscheidung
- beide gültig, aber uneinig → GLM-Adjudikation
- ein Primärgrader technisch ausgefallen, einer gültig → GLM muss mit dem gültigen Grader übereinstimmen
- beide Primärgrader technisch ausgefallen → unresolved
- GLM allein reicht nicht aus, wenn die Bewertungsregel zwei gültige Pfade verlangt

Damit verhindert man, dass ein dritter Grader einfach immer dann eingesetzt wird, wenn das Resultat unbequem ist.

---

9. Trennung von fachlichem Fehler und technischem Ausfall

AUTARK unterscheidet vier Ergebnisarten:

- **C — Correct**
- **I — Incorrect**
- **U — Unresolved**
- **T — Technical**

Diese Trennung ist zentral.

Ein technischer Ausfall ist nicht dasselbe wie eine falsche Antwort.

Beispiele für T:

- keine Finalantwort
- Request technisch gescheitert
- Modell läuft in ein definiertes Ausgabelimit
- Guard-/Runtimefehler verhindert verwertbare Antwort

Ein U bedeutet:

> *Es liegt eine Antwort vor, aber die eingefrorenen Bewertungsregeln erlauben keine hinreichend belastbare Entscheidung.*

Diese beiden Klassen werden **nicht im Nachhinein imputiert**.

Das verhindert insbesondere zwei Formen von Benchmarkkosmetik:

1. technische Ausfälle einfach aus dem Nenner entfernen;
2. unklare Fälle nachträglich als „wahrscheinlich richtig“ zählen.

---

10. End-to-End-Metrik versus bedingte Qualität

Gerade große Reasoning-Modelle können sehr starke Antworten erzeugen — **wenn sie fertig werden**.

Deshalb werden zwei verschiedene Fragen getrennt:

Bedingte Qualität

Wie gut ist das Modell auf den Fällen, in denen eine verwertbare Antwort vorliegt?

End-to-End-Leistung

Wie oft erzeugt das System unter realen Testbedingungen überhaupt eine korrekte, verwertbare Antwort?

RM16 zeigte genau, warum diese Unterscheidung wichtig ist.

R671:

- 13/15 der verwertbaren Antworten korrekt
- aber nur 15/24 technisch verwertbar

R70:

- geringere bedingte Korrektheit
- aber 20/24 technisch verwertbar

Für einen automatischen Router zählt End-to-End-Leistung, weil er vor dem Start nicht weiß, ob ein großes Modell diesmal fertig wird.

---

11. Prospektive Selektionsregeln

Eine Routingentscheidung wird nicht erst nach dem Test erfunden.

In RM16 wurde vor der Auswertung festgelegt, unter welchen Bedingungen ein automatischer R671-Selector überhaupt freigegeben werden dürfte.

Die Frozen Gates umfassten unter anderem:

1. genügend zusätzliche korrekte R671-Fälle,
2. mehr R671-only-C als R70-only-C,
3. höchstens einen zusätzlichen technischen Ausfall,
4. keine systematische Ressourcenstörung.

Nach dem Unblinding:

- R671-only C = 4
- R70-only C = 3
- aber R671 T = 9
- R70 T = 4

Damit scheiterte das Reliability-Gate klar.

Der automatische R671-Selector wurde deshalb **nicht freigegeben**.

Wichtig ist:

Die Freigaberegel wurde nicht nach Kenntnis des Ergebnisses gelockert.

---

12. Immutable closure

Wenn ein Benchmark oder eine experimentelle Stufe abgeschlossen ist, wird sie **geschlossen**.

Das bedeutet:

- keine nachträgliche Änderung des Scores,
- kein erneutes Grading gültiger Antworten,
- kein Austausch des Antwortschlüssels,
- kein Rerun derselben Author-Aufgabe, um ein besseres Resultat zu erhalten,
- keine nachträgliche Umklassifizierung eines unangenehmen Fehlers.

Ein geschlossenes Resultat bleibt historisch wahr, auch wenn man später mehr weiß.

---

13. Beispiel: der konventionssensitive φ³-Fall

Im finalen 48-Aufgaben-Test wurden offiziell vier Antworten als I klassifiziert.

Bei einem dieser Fälle — φ³-Theorie in sechs Dimensionen — zeigte eine spätere Post-Closure-Prüfung, dass der eingefrorene Antwortschlüssel konventionssensitiv beziehungsweise wahrscheinlich zu eng war.

Trotzdem blieb das offizielle Benchmarkresultat:

- **44 C**
- **4 I**
- **0 U**
- **0 T**

und nicht 45/48.

Das ist ein besonders anschauliches Anti-Goodhart-Beispiel:

> *Der Benchmark wird nicht nachträglich verbessert, nur weil man später einen guten Grund findet, einen alten Fehler anders zu beurteilen.*

Die spätere Analyse wird separat dokumentiert.

---

14. Append-only errata statt Geschichtsumschreibung

Auch technische Fehler in Testharnesses werden nicht aus der Geschichte gelöscht.

Wenn ein Runner einen Fehler enthält, entsteht:

- ein historischer Attempt,
- ein dokumentierter TECHNICAL_STOP,
- und ein neuer Successor beziehungsweise Erratum-Lauf.

Beispiel:

RM17.1 Attempt 1 stoppte im Preflight wegen eines Dictionary-Key-Mismatchs in der Runtime-Binding-Prüfung.

Das Resultat wurde nicht gelöscht oder umetikettiert.

Stattdessen wurde RM17.1.1 gebaut, das nur die identifizierte Harnessursache korrigierte.

So bleibt nachvollziehbar:

- was ursprünglich geschah,
- warum es geschah,
- und welche Änderung der Successor tatsächlich enthielt.

---

15. Minimal-change successors

Ein Erratum-Successor soll möglichst nur das reparieren, was nachweislich falsch war.

Beispiel RM17.1.1:

erlaubte Änderungen:

1. Key-Normalisierung der Runtime-SHA-Prüfung;
2. direkte Nutzung der kanonischen RM16.3.6-REQUEST-Dateien statt heuristischer Suche.

Nicht verändert wurden:

- wissenschaftliche Intervention,
- Taskset,
- Modelle,
- Retrysemantik,
- Ressourcenpolitik,
- Grading,
- Productionarchitektur.

Zusätzlich wurden unveränderte Runtime- und Resource-Funktionen per AST-Vergleich gegen den Parent geprüft.

Das reduziert die Gefahr, dass ein scheinbar technischer Fix unbemerkt zugleich das wissenschaftliche Experiment verändert.

---

16. Genau-einmal-Semantik

Für prospektive Author-Captures gilt häufig:

- genau ein Versuch pro Aufgabe,
- keine inhaltlichen Retries,
- keine stillen Wiederholungen schlechter Antworten.

Ein Retry kann sonst selbst ein Optimierungsmechanismus werden:

> *„Wenn die Antwort schlecht ist, fragen wir einfach noch einmal.“*

Ein System, das drei Versuche braucht, ist aber nicht dasselbe System wie eines, das die Aufgabe beim ersten Versuch löst.

Deshalb werden Wiederholungen nur dann zugelassen, wenn sie vorher ausdrücklich Teil des Protokolls sind.

---

17. Kein Salvage interner Reasoning-Tokens als Finalantwort

Bei Reasoning-Modellen kann es vorkommen, dass ein Modell sehr lange interne Reasoning-Ausgaben produziert, aber keine Finalantwort erreicht.

Diese Reasoning-Spur wird **nicht nachträglich als Antwort gerettet**.

Beispiel RM17:

Ein ausgewählter R671-Fall produzierte bei max_tokens=24576:

- sehr lange Reasoning-Ausgabe,
- finish_reason=length,
- aber keine Finalantwort.

Der Fall bleibt technisch unrecovered.

Das verhindert eine weitere Form nachträglicher Ergebnisverbesserung.

---

18. Failure-set diagnostics werden nicht als neue Benchmarks verkauft

Nach RM16 wurden neun technische R671-Ausfälle gezielt erneut untersucht.

Das ist bewusst ein **selected failure set**.

Die Frage lautet:

> *Wie viele dieser bekannten Ausfälle erholen sich, wenn nur der Completion-Budget-Parameter verändert wird?*

Das ist eine Mechanismusdiagnose.

Es ist **keine** Schätzung der allgemeinen R671-Reliability.

Deshalb wäre die Aussage

> *„1/2 recovered, also ist die Reliability 50 %“*

methodisch falsch.

Korrekt ist:

> *Von den ersten zwei gezielt ausgewählten früheren Failure-Fällen wurde einer unter dem höheren Budget technisch recovered, einer nicht.*

Diese Trennung zwischen:

- allgemeinem Benchmark,
- prospektivem matched comparison,
- und selektierter Failure-Diagnose

ist ein zentraler Anti-Goodhart-Bestandteil.

---

19. Keine Vermischung verschiedener Studien

RM15, RM16 und RM17 beantworten unterschiedliche Fragen.

RM15

Frage:

> *Wie gut funktioniert die eingefrorene Production-Architektur End-to-End auf einem frischen 48-Aufgaben-Holdout?*

RM16

Frage:

> *Würde ein automatischer R671-Pfad gegenüber R70 auf einem prospektiven matched Set genügend zusätzliche Leistung bringen, ohne zu viel Reliability zu verlieren?*

RM17

Frage:

> *Wie viel der beobachteten R671-Technical-Failure-Menge lässt sich durch ein höheres Completion-Budget erklären?*

Diese Resultate werden nicht zu einer einzigen Erfolgsquote zusammengerechnet.

---

20. Matched designs für Modellvergleiche

Wenn zwei Modelle verglichen werden, sollen sie nach Möglichkeit dieselben Aufgaben erhalten.

RM16 verwendete:

- 24 frische Aufgaben
- 12 Familien × 2 Varianten
- R70 und R671 auf denselben Aufgaben

Das erlaubt **paired analysis**.

Wichtiger als rohe Gesamtscores sind dann beispielsweise:

- beide korrekt
- nur R671 korrekt
- nur R70 korrekt
- beide nicht korrekt

Im geschlossenen RM16:

- beide C = 9
- R671-only C = 4
- R70-only C = 3
- neither C = 8

Der Netto-Unterschied war damit klein.

Der exakte gepaarte Test ergab:

- directional p = 0,5
- two-sided p = 1,0

Damit gab es keinen belastbaren statistischen Nachweis einer End-to-End-Überlegenheit von R671.

---

21. Cluster-Bootstrap statt Scheingenauigkeit

Zusätzlich zu exakten gepaarten Tests verwendet AUTARK Family-Cluster-Bootstraps.

Warum?

Wenn Aufgaben aus derselben Familie ähnliche Fehlerquellen teilen, darf man sie nicht behandeln, als wären sie vollständig unabhängige Stichproben.

Im finalen 48er-Test ergab der familiengeclusterte Bootstrap für die Trefferquote ein 95-%-Intervall von:

- **81,25 % bis 100 %**

Der Punktwert 91,67 % wird also nicht als universelle, hochpräzise Systemeigenschaft dargestellt.

---

22. Auditierbarkeit durch kryptographische Bindung

Wichtige Artefakte werden per SHA-256 gebunden.

Dazu gehören je nach Stufe:

- Tasksets
- Contracts
- Runner
- Blind-Corpora
- Gradingresults
- Closure-Artefakte
- Result-ZIPs
- Sourcebindings

Zusätzlich enthalten viele ZIPs ein internes SHA256SUMS.txt.

Damit kann später geprüft werden:

- ob genau dasselbe Artefakt vorliegt,
- ob Dateien verändert wurden,
- und ob ein Resultat tatsächlich zu dem behaupteten Freeze gehört.

Das ersetzt keine wissenschaftliche Methodik, erhöht aber die Nachvollziehbarkeit der Prozessgeschichte erheblich.

---

23. Source binding und Runtime identity

Bei Systemtests reicht es nicht, nur die Testaufgaben einzufrieren.

Auch die tatsächlich laufende Software muss identifiziert werden.

Dafür werden unter anderem SHA-256-Werte von:

- Production router,
- Start wrapper,
- Thermal guard,
- llama-server

geprüft.

Wenn ein erwarteter Sourcehash nicht stimmt, wird der Lauf abgebrochen.

So kann ein Resultat nicht unbemerkt mit einer anderen Runtime erzeugt werden als derjenigen, die im Testdesign vorgesehen war.

---

24. Ressourcenbedingungen sind Teil der Reproduzierbarkeit

Für große lokale Modelle sind Ressourcenbedingungen nicht bloß Operationsdetails.

Sie beeinflussen:

- technische Reliability,
- Laufzeit,
- mögliche Parallelität,
- Modellresidenz,
- und potenziell sogar das Verhalten des Gesamtsystems.

Deshalb werden bei Heavy-Modellen Bedingungen eingefroren, etwa:

- Swap muss vor Start 0 sein;
- R671 startet nur bei sehr hoher freier Speicherkapazität;
- andere Heavy-Modelle müssen kalt sein;
- Thermalzustände werden überwacht;
- mehrere Heavy-Modelle werden serialisiert;
- nach Ende muss ein Cold State nachgewiesen werden.

Ein wissenschaftlicher Test, der unter unsauberen Ressourcenbedingungen lief, wird nicht einfach so behandelt, als sei er mit einem kontrollierten Lauf identisch.

---

25. Kein kausaler Claim ohne Execution Evidence

AUTARK unterscheidet zwischen:

- Werkzeug ist installiert,
- Werkzeug ist integriert,
- Router plant Werkzeugnutzung,
- Gate akzeptiert Werkzeugnutzung,
- Werkzeug wurde tatsächlich ausgeführt,
- Werkzeugausführung hat die Finalantwort kausal beeinflusst.

Diese Stufen dürfen nicht miteinander verwechselt werden.

Im finalen 48-Aufgaben-Benchmark waren formale Werkzeuge integriert und die Toolarchitektur separat validiert.

Der Benchmark selbst bewies aber **keine tatsächliche CAS-/Lean-Ausführung für die relevanten Antworten**.

Deshalb wird aus 44/48 nicht gefolgert:

> *„Die formalen Werkzeuge haben den Score verbessert.“*

Das wäre ein kausaler Claim ohne ausreichende Execution Evidence.

---

26. Negative Evidenz bleibt sichtbar

Eine wichtige Anti-Goodhart-Regel lautet:

> *Ein negativer Befund wird nicht aus der Projektgeschichte entfernt, nur weil später ein besserer Successor existiert.*

Beispiele:

- fehlgeschlagene Harnessversuche bleiben dokumentiert;
- Technical Stops bleiben Technical Stops;
- ein verworfener R671-Selector bleibt verworfen;
- RM16 wird durch spätere RM17-Diagnostik nicht rückwirkend „repariert“;
- ein geschlossenes T bleibt im alten Experiment T, selbst wenn derselbe Task später unter anderen Bedingungen erfolgreich beantwortet wird.

Damit bleibt sichtbar, was das System **zu einem bestimmten historischen Zeitpunkt unter bestimmten Bedingungen tatsächlich geleistet hat**.

---

27. Warum Anti-Goodhart mehr ist als „kein Schummeln“

Goodharting muss nicht absichtlich geschehen.

Es entsteht oft gerade bei gewissenhafter Entwicklungsarbeit:

1. Man sieht einen Fehler.
2. Man versteht ihn besser.
3. Man verbessert das System.
4. Man testet dieselbe Aufgabe erneut.
5. Das Ergebnis ist besser.
6. Man beginnt unbewusst, diesen verbesserten Fall als Beleg für allgemeine Verbesserung zu behandeln.

Jeder einzelne Schritt ist vernünftig.

Der Fehler entsteht erst, wenn **Entwicklungsevidenz und unabhängige Evaluation verwechselt werden**.

Die AUTARK-V3-Methodik versucht deshalb, diese Kategorien institutionell auseinanderzuhalten.

---

28. Praktische Anti-Goodhart-Regeln in Kurzform

Die wichtigsten Regeln lassen sich so zusammenfassen:

1. **Freeze before exposure.**
2. **Fresh holdouts für entscheidende Leistungsclaims.**
3. **Keine nachträgliche Änderung gültiger Antwortschlüssel oder Scores.**
4. **C/I/U/T strikt getrennt halten.**
5. **Keine Imputation von U oder T.**
6. **Blind grading, wo möglich.**
7. **Mehrere Bewertungswege mit vorab festgelegter Adjudikation.**
8. **Keine stillen Retries.**
9. **Selected failure sets nicht als allgemeine Benchmarks interpretieren.**
10. **Geschlossene Experimente bleiben geschlossen.**
11. **Errata append-only statt historische Resultate umzuschreiben.**
12. **Minimal-change successors.**
13. **Runtime und Sourcehashes binden.**
14. **Kausale Toolclaims nur bei Execution Evidence.**
15. **Negative Ergebnisse sichtbar lassen.**
16. **Fresh prospective holdouts vor jeder neuen Productionfreigabe.**

---

29. Was diese Methodik nicht garantiert

Auch ein strenges Protokoll beseitigt nicht alle Unsicherheit.

Es verhindert nicht:

- kleine Stichproben;
- fehlerhafte Rubriken;
- Bewertungsfehler mehrerer Grader;

- unerkannte Benchmark-Leakage;
- Modellabhängigkeit der Grader;
- begrenzte Repräsentativität der Aufgaben;
- versteckte Korrelationen zwischen Taskfamilien;
- oder die Tatsache, dass akademische Aufgaben nur einen Ausschnitt realer Intelligenz abbilden.

Anti-Goodhart bedeutet deshalb nicht:

> *„Der Benchmark ist objektive Wahrheit.“*

Es bedeutet:

> ***Wir reduzieren systematisch die Möglichkeiten, uns selbst nach Kenntnis der Ergebnisse ein besseres System vorzutäuschen, als wir tatsächlich getestet haben.***

---

30. Der bisher stärkste praktische Test dieser Philosophie

Der aussagekräftigste Beleg für die Methodik ist nicht der Score selbst, sondern der Umgang mit unerwarteten Ergebnissen.

Drei Beispiele:

Beispiel A — 44/48 bleibt 44/48

Auch nachdem ein offizieller Fehler später als konventionssensitiv erkannt wurde, blieb der geschlossene Benchmark unverändert.

Beispiel B — kein automatischer R671-Selector

Obwohl R671 bei fertigen Antworten akademisch stark war, scheiterte es am vorher eingefrorenen Reliability-Gate. Der Selector wurde deshalb nicht freigegeben.

Beispiel C — RM17 ändert RM16 nicht

Wenn ein alter RM16-T unter 24576 Tokens später eine gute Finalantwort erzeugt, bleibt der alte RM16-Fall trotzdem T.

Der neue Lauf beantwortet eine **neue Frage unter neuen Bedingungen**.

---

31. Kurzform für eine öffentliche Beschreibung

> ***Umfang der Evidenz:*** *Der abschließende Score von 44/48 stammt aus einem frischen, vorab eingefrorenen End-to-End-Holdout. Er war nicht der gesamte Testumfang: First Pass und Extensions B/C/A umfassten zuvor bereits 288 kontrollierte author-seitige Evaluations-Slots/Runs; zusammen mit dem finalen Holdout, RM16 und den ersten RM17-Diagnosen sind bis zum hier dokumentierten Stand mindestens 386 author-seitige Evaluations-Slots/Runs explizit gezählt. Diese Zahl ist kein Benchmark-N unabhängiger Aufgaben und schließt Graderinferenzen, technische Retries und weitere Kalibrations-/Safety-Tests nicht ein.*
>
> ***AUTARK Academic V3 verwendet eine bewusst Anti-Goodhart-orientierte Testmethodik.*** *Entscheidende Tests werden vor der Ausführung eingefroren: Aufgaben, Reihenfolge, Bewertungsschlüssel, Gradingregeln, technische Failure-Kriterien und gegebenenfalls Freigabeschwellen. Für wichtige Leistungsclaims werden frische Holdouts verwendet. Antworten werden soweit möglich blind und über mehrere Bewertungswege beurteilt; technische Ausfälle, fachliche Fehler und ungelöste Fälle bleiben getrennt. Geschlossene Benchmarks werden nachträglich nicht umgeschrieben — auch dann nicht, wenn eine spätere Analyse einen alten Fall günstiger erscheinen lässt. Diagnostische Wiederholungen bekannter Fehlerfälle werden als neue Experimente behandelt und nicht zur rückwirkenden Verbesserung alter Scores benutzt. Artefakte, Runner und Resultate*

werden kryptographisch gebunden, und kausale Aussagen über Tools werden nur gemacht, wenn ihre tatsächliche Ausführung nachgewiesen ist.

---

32. Ein-Satz-Version

Die Anti-Goodhart-Idee lautet: Erst festlegen, was als Erfolg zählt; dann messen; danach das historische Ergebnis nicht mehr passend machen — sondern jede Verbesserung in einem neuen, getrennten Experiment beweisen.

---

33. Autoritative Beispiele aus dem bisherigen Projekt

Finaler 48-Aufgaben-Benchmark

Status:

RM151228_PASS_FINAL_BENCHMARK_CLOSED_NO_FURTHER_BENCHMARK_INFERENCE_OR_REGRADE

Resultat:

- C44
- I4
- U0
- T0
- N48
- technische Reliability 100 %

Closure SHA-256:

2f50add048b0bec7d7ea560b7b5fcc6bf4f81916ba39e11282e9760612378e62

RM16 matched R70-vs-R671

Resultat:

- R70: C12 / I5 / U3 / T4
- R671: C13 / I2 / U0 / T9
- automatic R671 selector: REJECT

Closure SHA-256:

0d825df9c98fee8c154570d1c7202c4e27d442bed3af6b3311ef6a3d41e42054

RM17.1.1A selected-failure diagnostic

Zwei der neun eingefrorenen alten R671-T-Fälle wurden unter einem höheren Completion-Budget untersucht:

- einer technisch recovered,
- einer bei 24576 Tokens weiterhin finish_reason=length.

Diese Resultate ändern RM16 nicht.

Result SHA-256:

4a175a8e3b5974b1c9d9247b081d2e76d20395b0048a07072d34bc2a749043af

Schluss

Die Testmethodik von AUTARK V3 ist bewusst konservativ.

Sie versucht nicht, jede Unsicherheit durch eine einzige Kennzahl zu beseitigen.

Stattdessen trennt sie:

- Entwicklung von Evaluation,
- technische Reliability von fachlicher Korrektheit,
- allgemeine Benchmarks von ausgewählten Diagnosen,
- beobachtete Korrelation von kausaler Evidenz,
- und historische Resultate von späteren Verbesserungen.

Ein weiterer Grundsatz betrifft die Darstellung des **Testumfangs**: große Call- oder Run-Zahlen dürfen nicht als Zahl unabhängiger Aufgaben ausgegeben werden. Umgekehrt darf der finale frische Holdout nicht so dargestellt werden, als habe das System davor nur 48 Tests gesehen.

Der wichtigste Anti-Goodhart-Grundsatz lautet dabei:

> ***Eine Verbesserung zählt erst dann als allgemeiner Fortschritt, wenn sie sich in einem neuen, vorab festgelegten und vom ursprünglichen Fehler möglichst unabhängigen Test bestätigt.***