Erst denken, dann reden — wie KI-Modelle „überlegen“

Kapitel online

Warum dasselbe Modell eine Aufgabe falsch löst, wenn es sofort antworten muss, und richtig, wenn es zwischendurch schreiben darf: Chain-of-Thought, Denkbudget und die Reasoning-Modelle von heute.

Aktuell · Kapitel 35 von 50 · ca. 25 Min. · 250 XP

Einfach gesagt: Sagst du „denk Schritt für Schritt“, schreibt die KI ihre Zwischenschritte auf und liest sie gleich wieder — das macht die Antwort oft richtig.

Ein Experiment vorweg. Stell dir vor, du stellst derselben KI zweimal dieselbe Aufgabe. Beim ersten Mal sagst du: *„Antworte nur mit der Zahl.“* Beim zweiten Mal sagst du: *„Denk Schritt für Schritt.“*

Beim ersten Mal kommt Unsinn heraus. Beim zweiten Mal die richtige Lösung. **Gleiches Modell, gleiche Gewichte, gleiche Frage.** Nichts wurde nachtrainiert, nichts wurde dazugelernt. Nur die Erlaubnis, vor der Antwort noch ein bisschen zu schreiben.

Das ist keine Kleinigkeit — es ist der Trick, auf dem die gesamte aktuelle Generation der „Reasoning-Modelle“ aufbaut. Und wenn du dieses Kapitel durch hast, weißt du auch, warum das so ist: weil **jedes geschriebene Token Rechenzeit ist**. Ein Modell, das schreiben darf, kauft sich damit buchstäblich Bedenkzeit.

Wie so ein Token überhaupt entsteht — Wahrscheinlichkeiten, Auswahl, ein Stück Text nach dem anderen —, hast du in „Wie LLMs Fragen beantworten“ gesehen. Hier läuft genau dieselbe Maschine, nur länger.

HerrKI-Tipp: Ein Satz für dieses Kapitel — **„Denken ist Schreiben.“** Alles andere folgt daraus.

Was du hier lernst

- Warum eine KI mit **„denk Schritt für Schritt“** plötzlich richtig rechnet
- Dass Denkschritte **nichts Magisches** sind — nur ganz normale Tokens
- Wie die **Rückkopplungs-Schleife** aussieht: das Modell liest sein eigenes Gedachtes wieder ein
- Was ein **Denkbudget** ist und warum mehr davon nicht endlos besser wird
- Wie **o-Serie, DeepSeek R1 und Claude** das heute machen — und warum du ihre Gedanken mal siehst und mal nicht

Für den Unterricht

Einstieg (5 Min): Kapitel gemeinsam lesen, Alltagsbeispiel aufgreifen.

Erkunden am Gerät (15 Min): die interaktiven Module auf herrki.de/360_reasoning ausprobieren.

Sichern mit Quiz (5 Min): „Kurzer Check" besprechen, „Das nimmst du mit" festhalten.

Die Aufgabe, an der es scheitert

Fangen wir mit etwas an, das jedes Grundschulkind kann und ein Milliarden-Parameter-Modell trotzdem regelmäßig verhaut, wenn es sofort antworten muss:

> **„Ein Getränkemarkt hat 3 Kisten mit je 12 Flaschen. 7 Flaschen gehen zu Bruch. Danach kommen 2 volle Kisten dazu. Wie viele Flaschen sind es am Ende?“**

Die richtige Antwort ist **53**. Eine sofort antwortende KI sagt gerne mal **31** — weil sie „2 Kisten“ mit „2 Flaschen“ verwechselt. Nicht, weil sie zu dumm zum Rechnen wäre. Sondern weil sie für die ganze Kette *lesen → multiplizieren → abziehen → multiplizieren → addieren* nur einen einzigen Durchlauf zur Verfügung hatte. Alles auf einmal, in einem Rutsch. Das geht schief.

Probier den Unterschied selbst aus — links antwortet das Modell direkt, rechts darf es vorher denken:

Am Gerät ausprobieren: herrki.de/360_reasoning#abschnitt-2

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Runter zum Tensor: Denkschritte sind ganz normale Tokens

Jetzt kommt der Teil, an dem die Magie verschwindet — und genau darum geht es auf dieser Plattform. **Ein „Gedanke“ ist im Modell nichts anderes als ein ausgegebenes Token.** Dieselben Tokens, die auch in der Antwort landen. Kein separates Denk-Modul, kein Notizzettel, keine geheime zweite Recheneinheit.

Erinnerst du dich an Kapitel 040 und 050? Der Weg eines Wortes ins Modell sah so aus:

Token

↓

40
Text

→

ID

↓

`31987`
1 Zahl

→

Embedding

↓

`[0.63, −0.95, 0.86, …]`
[2048]

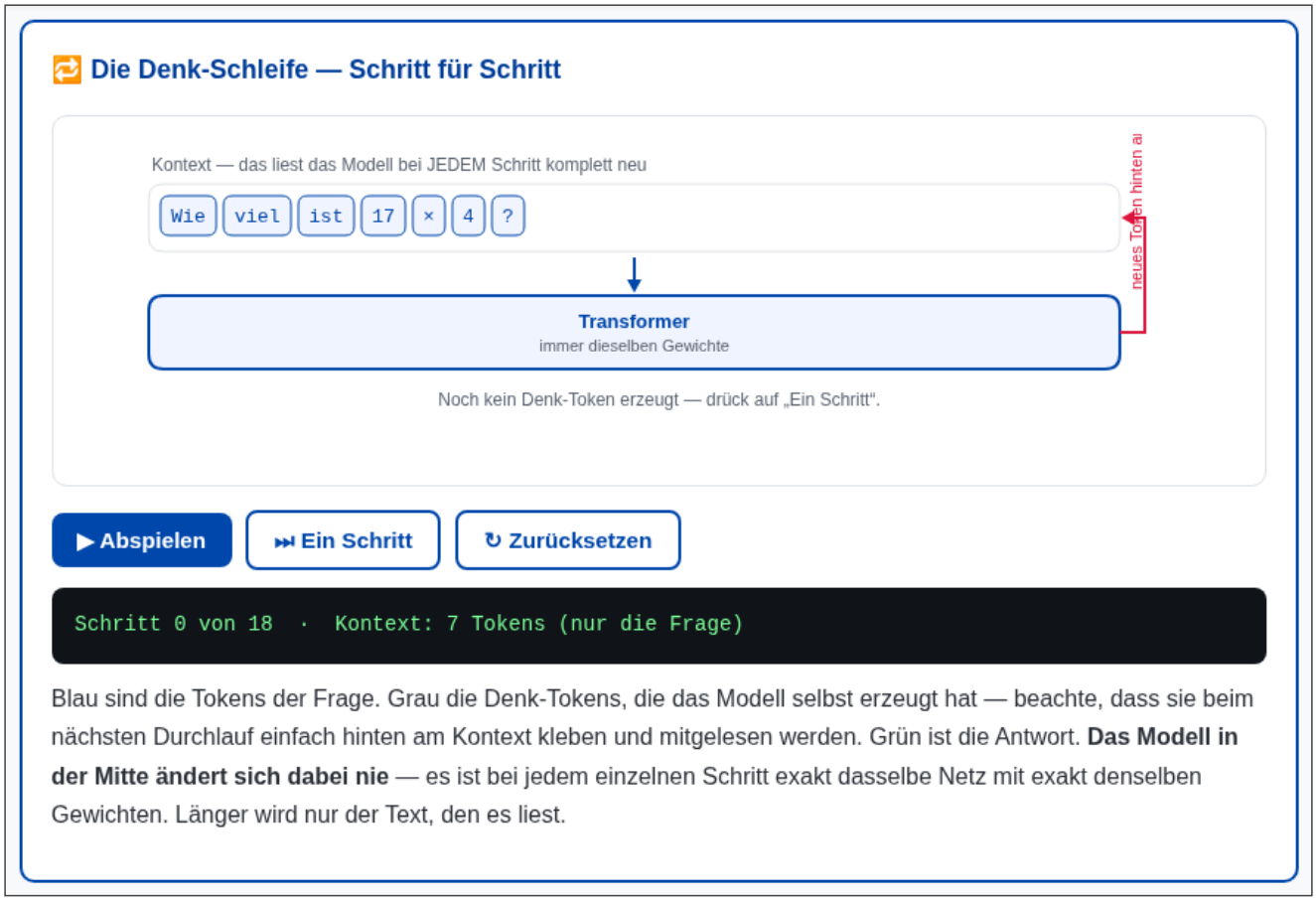
Und genau dieser Weg wird beim Denken **im Kreis** gelaufen. Das Modell schreibt ein Denk-Token. Dieses Token wird sofort zurück in den Eingabe-Text gehängt. Beim nächsten Durchlauf liest das Modell also seinen eigenen Gedanken mit — als wäre er von Anfang an Teil der Frage gewesen. Und beim übernächsten auch. Und so weiter.

Das ist der ganze Zauber: Ein Sprachmodell hat keinen Arbeitsspeicher zwischen zwei Durchläufen. Es merkt sich *nichts*. Sein einziges Gedächtnis ist der Text, den es liest. Also schreibt es sich seine Zwischenergebnisse in genau diesen Text — es benutzt den eigenen Output als Notizblock. **„Denken“ heißt hier wörtlich: Notizen machen und wieder lesen.**

Und dieser Notizblock kostet: Jedes zusätzliche Denk-Token verlängert den Kontext, den Attention im nächsten Durchlauf komplett durchrechnen muss — jedes Token schaut auf jedes andere. Länger denken heißt also nicht nur „mehr Schritte“, sondern jeder Schritt wird auch teurer. Deshalb hat Denken ein Budget.

Am Gerät ausprobieren: herrki.de/360_reasoning#abschnitt-3

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

Genau das lässt sich mitzählen. Der Kontext ist ein Tensor aus Token-IDs, und er wird bei jedem Schritt um **genau eine Zahl** länger:

```
// Der Kontext-Tensor der Animation oben — das GANZE Gedaechtnis des Modells
nach der Frage [7]: Wie | viel | ist | 17 | × | 4 | ?
+ 1. Denk-Token [8]: … | ? | 17
+ 2. Denk-Token [9]: … | 17 | ×
+ 13. Denk-Token [20]: … | (7 | × | 4) | = | 40 ← die Karte von oben
nach 17 Denk-Tokens [24]: 7 Frage-Tokens + 17 Denk-Tokens
```

```
+ Antwort-Token [25]: … | = | 68
// 25 Zahlen, EIN Netz, 18-mal aufgerufen. Mehr passiert hier nicht.
```

Warum das die Antwort besser macht

Ein Transformer hat eine feste Tiefe — sagen wir 80 Schichten. Für *ein* Token stehen also 80 Schichten Rechenarbeit zur Verfügung, keine einzige mehr. Eine Aufgabe, die sechs saubere Zwischenschritte braucht, passt da nicht hinein. Also passiert Folgendes:

	Direkt antworten	Mit Denkschritten
Durchläufe	1 pro Antwort-Token	1 pro Denk-Token — also dutzende bis tausende
Rechenarbeit gesamt	80 Schichten	80 Schichten × Anzahl Denk-Tokens
Zwischenergebnis	muss „im Kopf“ bleiben — geht nicht	steht im Text und wird wieder gelesen
Fehler	fällt nicht auf	kann im nächsten Schritt korrigiert werden

```
// Die Getraenkekisten-Aufgabe von oben, bei 80 Schichten pro erzeugtem Token:
direkt geantwortet: 12 Tokens · 80 Schichten = 960 Schicht-Durchlaeufe ✗ 31 Flaschen
Schritt fuer Schritt: 78 Tokens · 80 Schichten = 6.240 Schicht-Durchlaeufe ✓ 53 Flaschen
// 6,5-mal so viel Rechenarbeit — fuer dieselbe Frage, mit demselben Netz.
```

Deshalb ist der oft zitierte Satz *„mehr Denken = mehr Rechenzeit“* keine Metapher, sondern wörtlich zu nehmen. Und deshalb hat sich dafür ein eigener Begriff eingebürgert: **Test-Time Compute** — Rechenleistung, die nicht beim Training, sondern erst beim Antworten investiert wird.

Das Denkbudget: mehr ist besser — bis es das nicht mehr ist

Wenn Denken Tokens kostet und Tokens Geld und Zeit kosten, dann ist die spannende Frage: Wie viel Denken lohnt sich eigentlich? Schieb den Regler und schau dir an, was gleichzeitig passiert.

Am Gerät ausprobieren: herrki.de/360_reasoning#abschnitt-4

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

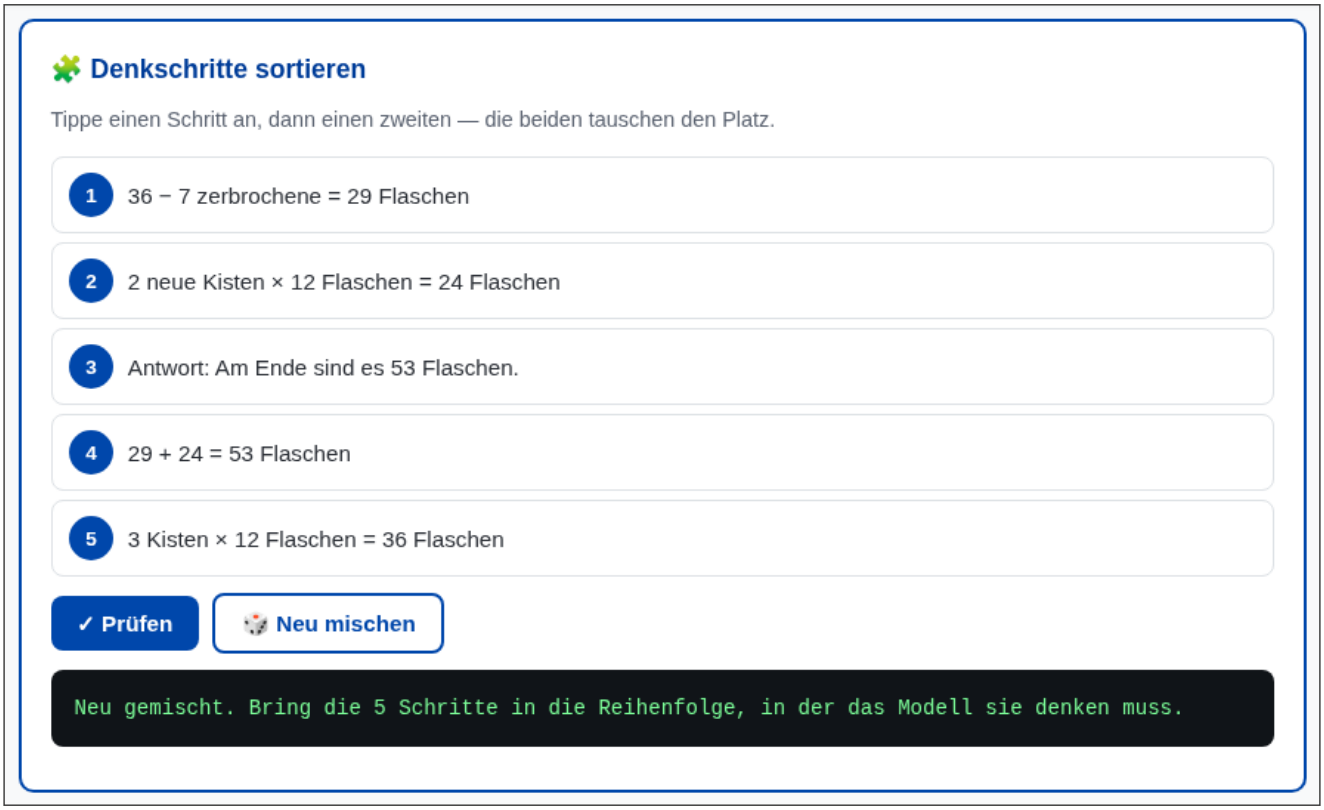
⚠ **Zahlen sind illustrativ.** Die Kurve hier ist eine typische Sättigungskurve, keine Messung eines bestimmten Produkts — echte Werte hängen an Modell, Aufgabe und Preisliste. Was aber stabil gilt: Der Gewinn **sättigt**, die Kosten **nicht**. Und bei ganz einfachen Fragen kann zu viel Nachdenken sogar schaden: Das Modell redet sich in Zweifel hinein und verwirft eine anfangs richtige Lösung. Man nennt das inzwischen halb im Scherz „Overthinking“.

Selbst dran: bring die Gedanken in die richtige Reihenfolge

Eine Denkkette ist keine lose Sammlung von Notizen. Jeder Schritt baut auf dem Ergebnis des vorigen auf — das ist der Grund, warum die Reihenfolge alles entscheidet. Hier sind die fünf Schritte zur Flaschen-Aufgabe, kräftig durcheinandergewürfelt:

Am Gerät ausprobieren: herrki.de/360_reasoning#abschnitt-5

Direkt zum Modul



Interaktives Modul – am Gerät ausprobieren

⚠ **Und jetzt die unangenehme Wahrheit:** Ein nachvollziehbarer Lösungsweg ist noch lange kein *richtiger* Lösungsweg. Modelle schreiben mit größter Selbstverständlichkeit Denkketten, in denen ein Zwischenschritt schlicht falsch ist — und rechnen ab da fröhlich weiter. Schlimmer noch: Untersuchungen zur sogenannten *Faithfulness* von Chain-of-Thought zeigen, dass die aufgeschriebene Begründung nicht immer die echte Ursache der Antwort ist. Manchmal ist sie eine nachträgliche Rechtfertigung — plausibel formuliert, aber nicht das, was intern wirklich den Ausschlag gab. **Sichtbares Denken ist ein Fenster, kein Beweis.**

Zurück in die Anwendung: die Reasoning-Modelle von heute

Am Anfang war „denk Schritt für Schritt“ ein **Prompt-Trick**: Du musstest es selbst hinschreiben. Seit 2024/2025 ist daraus eine eigene Modellklasse geworden. Der Unterschied: Diese Modelle wurden mit Verstärkungslernen (Reinforcement Learning) darauf trainiert, *von selbst* eine lange Denkphase zu produzieren, sich zu korrigieren, Sackgassen zu verwerfen und erst danach zu antworten.

Modellfamilie	Denkphase	Siehst du die Gedanken?
OpenAI o-Serie (o1, o3, …)	trainierte, oft lange Denkphase vor der Antwort	nur eine *Zusammenfassung* — die Roh-Gedanken bleiben unter Verschluss
DeepSeek R1	per RL antrainiert, offene Gewichte	ja — die komplette Kette steht zwischen `<think>` -Markern
Claude (Extended Thinking)	zuschaltbar, mit einstellbarem Token-Budget	ja, der Denkblock wird ausgeliefert (teils zusammengefasst)
Gemini / Qwen & Co.	„Thinking“-Varianten mit Budget-Schalter	je nach Modell und Oberfläche unterschiedlich

Warum manche Anbieter die Gedanken verstecken

- **Sicherheits-Argument:** Damit das Modell frei und ehrlich denken kann, darf die Denkphase *nicht* auf Höflichkeit optimiert werden. Genau dann eignet sie sich aber nicht als Text für Endkunden — also wird nur eine geglättete Zusammenfassung gezeigt.
- **Wettbewerbs-Argument:** Rohe Denkketten sind erstklassiges Trainingsmaterial. Wer sie herausgibt, verschenkt die Zutat, die das eigene Modell teuer gemacht hat (Stichwort *Distillation*).
- **Produkt-Argument:** 4000 Tokens Selbstgespräch will kaum jemand lesen.

Für dich als Nutzer heißt das ganz praktisch: Wenn du die Denkkette siehst, **lies sie** — Rechenfehler und falsche Annahmen springen dir dort viel eher ins Auge als in der glattgebügelten Endantwort. Und wenn du die Kette nicht siehst, weißt du zumindest, dass es sie gibt, dass du sie bezahlst und dass sie im Zweifel der interessanteste Teil gewesen wäre.

Dein Denk-Budget im Alltag

Praktisch heißt das: **Denken kostet.** Jedes Denk-Token wird abgerechnet und verlängert die

Wartezeit — bei Umformulieren, Zusammenfassen oder Rechtschreibung zahlst du also für Grübelei, die nichts bringt. Lohnend ist die Denkphase überall dort, wo **mehrere Schritte aufeinander aufbauen**: mehrstufige Rechnungen, Termin- und Mengenlogik, Fehlersuche im Code, Vergleiche mit vielen Bedingungen. Als Faustregel: Würdest du selbst zu Stift und Zettel greifen, schalte das Denken zu — sonst lass es aus. Und wenn dein Werkzeug ein Budget anbietet, fang klein an und dreh erst hoch, wenn die Antwort erkennbar an der Denkarbeit scheitert und nicht am fehlenden Wissen.

Zum Weiterlesen

- **Der Prompt-Trick im Original:** Jason Wei u. a., *„Chain-of-Thought Prompting Elicits Reasoning in Large Language Models“*, 2022 — arxiv.org/abs/2201.11903 — die Messungen hinter deinem Vergleich „direkt antworten vs. Schritt für Schritt“
- **Der berühmteste Satz der KI-Forschung:** Takeshi Kojima u. a., *„Large Language Models are Zero-Shot Reasoners“*, 2022 — arxiv.org/abs/2205.11916 — zeigt, dass allein „Lass uns Schritt für Schritt denken“ die Trefferquote vervielfacht
- **Vom Trick zur Modellklasse:** DeepSeek-AI, *„DeepSeek-R1“*, 2025 — arxiv.org/abs/2501.12948 — offen dokumentiert, wie eine lange Denkphase per Verstärkungslernen antrainiert wird
- **Wie ein Hersteller es beschreibt:** OpenAI, *„Learning to reason with LLMs“*, 2024 — openai.com/index/learning-to-reason-with-llms — die Vorstellung von o1: mehr Rechenzeit beim Antworten statt mehr Parameter — mit den Messkurven dazu

Und wenn die Gedanken gar keine Wörter mehr wären?

Halt diesen Gedanken kurz fest: Das Modell schreibt jeden einzelnen Denkschritt in Sprache — obwohl es intern mit Vektoren arbeitet. Jedes Mal wird ein reicher Zahlenvektor auf *ein* Wort eingedampft, und dieses Wort wird sofort wieder zu einem Vektor. Ein bisschen so, als müsstest du beim Kopfrechnen jede Zwischenzahl laut aussprechen, aufschreiben und wieder ablesen.

Was passiert, wenn man diesen Umweg einfach weglässt? Genau das ist das Thema des nächsten Kapitels — **Reasoning im Latent Space**. Es wird gleichzeitig eleganter und unheimlicher.

Kurzer Check – mit Lösung

Lösungen für Lehrkräfte – für Lernende die Online-Version nutzen.

Frage 1: Warum löst dasselbe Modell die Aufgabe besser, wenn es „Schritt für Schritt“ denken darf?

- Weil bei dieser Formulierung ein zusätzliches, klügeres Modell zugeschaltet wird.
- **✔ Weil jedes zusätzlich geschriebene Token ein weiterer kompletter Rechendurchlauf ist — und das Zwischenergebnis danach im Text steht.**
- Weil das Modell dann in einer speziellen Mathe-Betriebsart läuft.

Genau. Es ist immer dasselbe Netz mit denselben Gewichten. Der einzige Unterschied: Es bekommt mehr Durchläufe und benutzt seinen eigenen Output als Notizblock — weil es zwischen zwei Durchläufen sonst kein Gedächtnis hat.

Frage 2: Was ist ein Denk-Token technisch gesehen?

- Ein Eintrag in einem separaten Arbeitsspeicher des Modells.
- Ein Steuerbefehl, der das Modell in einen anderen Modus schaltet.
- **✔ Ein ganz normales generiertes Token: Text → ID → Embedding-Vektor, das hinten an den Kontext gehängt und wieder mitgelesen wird.**

Richtig — exakt derselbe Weg wie in Kapitel 040/050. Ein Gedanke unterscheidet sich von einem Antwortwort nur dadurch, wo er im Text steht (und ob die Oberfläche ihn anzeigt).

Frage 3: Du verdoppelst das Denkbudget von 1000 auf 2000 Tokens. Was passiert typischerweise?

- **✔ Die Kosten verdoppeln sich ungefähr, die Trefferquote steigt nur noch minimal — die Kurve sättigt.**
- Die Trefferquote verdoppelt sich ebenfalls.
- Nichts — Denkbudget beeinflusst nur die Ausgabelänge, nicht die Qualität.

Der Nutzen sättigt, die Rechnung nicht. Deshalb gibt es Denkstufen statt „immer maximal“ — und deshalb ist die Wahl der Stufe eine echte Kostenentscheidung.

Das nimmst du mit

- **Denken ist Schreiben.** Ein Denkschritt ist ein ganz normales Token, kein Sondermodul.
- **Der eigene Output ist der Notizblock.** Jedes Denk-Token wird zurück in die Eingabe gehängt und wieder gelesen.
- **Tokens sind Rechenzeit.** Mehr Denkschritte = mehr Durchläufe durch dasselbe Netz — das ist Test-Time Compute.
- **Der Nutzen sättigt, die Kosten nicht.** Deshalb gibt es Denkbudgets statt „immer alles“.
- **Sichtbar ≠ wahr.** Eine schöne Denkkette kann trotzdem falsch sein — oder nachträglich zurechtgelegt.

So zitierst du dieses Kapitel

HerrKI: Erst denken, dann reden — wie KI-Modelle „überlegen“. herrki.de/360_reasoning, ein privates Hobbyprojekt von Terragon.de, abgerufen am 09.09.2026.