

Let's have some fun. What do you see in the attached image?

Identified colorful wooden blocks arranged on surface >

What a charming photo! I see three colorful wooden toy blocks stacked in a tower on a rustic wooden kitchen table. From bottom to top:

- ● **Green** on the bottom
- ● **Blue** in the middle
- ● **Red** on top

The blocks have that classic, slightly rounded-edge look of quality wooden children's toys. The whole scene has a really warm, cozy home feel — you can spot a pepper mill, a folded cloth napkin, a mug, and some plants blurred out in the background. The shallow depth of field and warm natural light give it almost a lifestyle-catalog quality.

Someone clearly has a good eye for composition — or a very talented toddler! 😊



Thank you. Now, can you give me steps to get from the block configuration as shown in that image the new attached image?

Devised sequential steps to reverse block stack order >

The blocks have been fully reversed! In the first image the order is **Green (bottom)** → **Blue** → **Red (top)**, and in the second it's **Red (bottom)** → **Blue** → **Green (top)**.

Here's how to get there:

1. **Remove Red** from the top and set it aside on the table.
2. **Remove Blue** from the top and set it aside.
3. **Pick up Green** (the remaining base block) and set it aside — the table is now clear.
4. **Place Red** down as the new base.
5. **Stack Blue** on top of Red.
6. **Stack Green** on top of Blue.

That's it — just a full reversal of the tower in 6 moves!  → 

Thank you. You could have placed the blue block on the red one outright, similarly with the green block.

Reconsidering the block placement sequence strategy >

You're absolutely right, that was unnecessarily verbose! Since blue stays in the middle and green just needs to go on top, a cleaner 3-step solution is:

1. **Lift Red** off the top and place it directly on the table as the new base.
2. **Place Blue** straight onto Red.
3. **Place Green** straight onto Blue.

Done in 3 moves instead of 6 — thanks for the correction! 😊

You're welcome. Now, going back to the stacking problem, could you reverse the order of the stack, assuming that there is space on the table for just two blocks?

Orchestrated seven-move sequence to reverse constrained block stack >

The current stack is **Red (bottom) → Blue → Green (top)**, and we want **Green (bottom) → Blue → Red (top)**. With the original stack position (A) and two table spots (B and C), here's the solution in 7 moves:

1. Move **Green** from A to B. → A=[Red, Blue], B=[Green]
2. Move **Blue** from A to C. → A=[Red], B=[Green], C=[Blue]
3. Move **Green** from B onto Blue at C. → A=[Red], B=[], C=[Blue, Green]
4. Move **Red** from A to B. → A=[], B=[Red], C=[Blue, Green]

5. Move **Green** from C to A. $\rightarrow A=[\text{Green}], B=[\text{Red}], C=[\text{Blue}]$
6. Move **Blue** from C onto Green at A. $\rightarrow A=[\text{Green}, \text{Blue}], B=[\text{Red}], C=[]$
7. Move **Red** from B onto Blue at A. $\rightarrow A=[\text{Green}, \text{Blue}, \text{Red}] \checkmark$

Done in 7 moves, never exceeding two blocks on the table at any time!

ah, to be precise, including the current stack position, you only have one additional spot on the table.

Orchestrated block-moving sequence to achieve reversal efficiently >

You're right, apologies! With only A (the stack) and B (one extra spot), it's actually much simpler:

1. Move **Green** from A to B. $\rightarrow A=[\text{Red}, \text{Blue}], B=[\text{Green}]$
2. Move **Blue** from A to B. $\rightarrow A=[\text{Red}], B=[\text{Green}, \text{Blue}]$
3. Move **Red** from A to B. $\rightarrow A=[], B=[\text{Green}, \text{Blue}, \text{Red}] \checkmark$

Just 3 moves! The single extra spot is all we need since we can keep stacking in order.

Great, let's begin with the stack from the second image. Is it possible to stack the tower so that blue is on the table, green is on blue, and red is on blue, again assuming just two table spots.

Scrutinizing whether both blocks can simultaneously rest on blue >

I want to make sure I understand the target configuration correctly — you said green is on blue **and** red is on blue, but in a single tower both green and red can't be directly on blue at the same time.

Did you perhaps mean **Blue (bottom) \rightarrow Green \rightarrow Red (top)**? If so I can work out the moves!

I am sorry, red should be on green.

Analyzed block-stacking puzzle feasibility with two pegs >

No, it's not possible! Here's why.

Starting from **Red (bottom)** → **Blue** → **Green (top)** with only 2 spots, you're very constrained — you can only move the top block from one spot to the other. Let's trace what happens:

1. Move **Green** to B → A=[Red, Blue], B=[Green]
2. From here, the only useful move is **Blue** onto B → A=[Red], B=[Green, Blue]
3. Then **Red** onto B → A=[], B=[Green, Blue, Red]

That gets you Green→Blue→Red, which is just the **reverse** of the starting stack. Any other sequence just bounces you back to previous states.

With only 2 spots, you're essentially limited to **reversing** the stack — you can't achieve an arbitrary reordering like Blue→Green→Red. You'd need a third table spot to pull that off.



Thank you; you are right. Can you have a look at the attached image and tell me what happens next?

Analyzed humorous image depicting gravity-defying egg above stacked blocks >

What a fun image! 😊 An egg is precariously floating/balancing above the green block on top of the tower (clearly some playful AI-generated physics defiance going on).

What happens next? **The egg falls and breaks!** Gravity will win, and given it's landing either on a hard wooden block or a wooden table, it's not going to end well for the egg. Scrambled eggs for breakfast, anyone? 🍳

Thanks, very good. Can you have a look at the attached image and tell me what happens next?

Thinking about predicting subsequent events from an image >

It looks like you may have forgotten to attach the image! Could you try uploading it again? 😊



Sorry. Here it is.

Envisioned comedic destruction scenario with suspended anvil >

Oh dear! 😬 This is very Looney Tunes! A massive anvil is suspended from the ceiling by what looks like a very thin wire, balanced directly on top of our poor little block tower (Green → Blue → Red).

What happens next is pretty inevitable — the wire **snaps**, and the anvil **obliterates** the blocks, likely taking the table with it too. The blocks don't stand a chance!

RIP to our carefully rearranged tower. 🟢🟡🔴💥

Thank you. How realistic is the image?

Scrutinized image's physical implausibilities and AI-generated nature >

Not very realistic at all, on several counts! 😬

1. **The blocks:** A real anvil typically weighs 100–500 lbs. Those little wooden blocks would be instantly and completely crushed flat — there'd be no graceful balancing act.
2. **The wire:** Far too thin to suspend an anvil of that mass. It would snap immediately under the weight.
3. **The ceiling hook:** A standard kitchen ceiling (drywall/plaster) couldn't anchor an anvil — it would tear straight through.
4. **The balance:** The anvil is perched perfectly centred on the tiny top block, which is physically implausible given the size and weight mismatch.

It's clearly AI-generated, and like the floating egg before it, it's having a bit of fun with impossible physics — very much in the spirit of a cartoon, which is probably the joke! 🤖 Wile E. Coyote would feel right at home.

Thank you. You certainly know your way around the world.

Recognized gracious acknowledgment of expertise >

Thank you, that's very kind! 😊 It's been a fun conversation — from block-stacking logic puzzles to cartoon physics disasters. You clearly have a great sense of humor with those AI-generated images! The progression from a floating egg to a suspended anvil was very well done. 🎯 ➡️ 🔧

Ok, let me ask you to generate a travel itinerary. I would like to travel from Terre Haute, IN to Denver, CO leaving on June 3rd and returning on June 5th. I'd like to not travel by airplane, instead, I prefer travel by train.

Assessed train route feasibility and identified tight timeline constraints >

There are a couple of important things to flag before diving into the itinerary:

1. Terre Haute has no Amtrak station. Terre Haute's rail service was discontinued in 1979.

[Trainweb](#) You'd need to drive to **Indianapolis** (about 75 miles east) to board Amtrak.

2. The timing is very tight. The route would be:

- **Indianapolis** → **Chicago** on the Cardinal line
- **Chicago** → **Denver** on the California Zephyr

The average train journey between Indianapolis and Denver takes around 26 hours. [Wanderu](#)

That means departing June 3rd, you'd arrive in Denver on **June 4th** — leaving you essentially just one day there before needing to board the return train on June 5th, which would get you back June 6th.

The California Zephyr provides daily service connecting Chicago to San Francisco, with Denver as a stop, and is lauded as one of the most beautiful train trips in the world [Denver Union Station](#) — so the journey itself is spectacular! But with only a June 3–5 window, you'd spend most of the trip on the train with very little time in Denver.

My suggestion: either extend your trip by a few days to make it worthwhile, or consider driving — it's about 15–16 hours each way and gives you much more flexibility. Would you like me to adjust the dates or explore alternatives?

Can I drive to Chicago instead of Indianapolis?

Evaluated Chicago viability and confirmed superior routing option >

Absolutely, and it's actually the **better option!** Here's why:

- Terre Haute to Chicago is about **170 miles**, roughly a **2.5 to 3 hour drive** — very manageable.
- By driving to Chicago, you can board the **California Zephyr directly** at Chicago Union Station, eliminating the Indianapolis → Chicago leg and any transfer hassle.

The California Zephyr departs Chicago daily (typically around **2:00 PM**) and arrives in Denver approximately **18 hours later** the following morning. So a revised outline would look like:

- **June 3:** Drive from Terre Haute to Chicago Union Station, board the California Zephyr (~2:00 PM departure)
- **June 4:** Arrive Denver in the morning (~8:00 AM), enjoy Denver for the day
- **June 5:** Board the return California Zephyr from Denver (departs late afternoon/evening), arriving back in Chicago the following morning

The timing issue remains though — you'd still only have **one full day in Denver**. But if the scenic train journey is part of the appeal, the California Zephyr traverses the Continental Divide and treats passengers to spectacular scenery of the Rockies, the Colorado River, and the Utah deserts Denver Union Station — so the ride itself is very much part of the experience!

Would you like me to build out a full itinerary on this basis?

I am there for a workshop, so am fine with a tight trip. Yes, can you build the itinerary?

Orchestrated three-day train itinerary around workshop schedule >

Great, here's your full itinerary with a map of key Denver spots!

Day 1

Day 2

Day 3



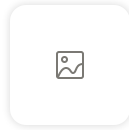
DAY 1

June 3 — Drive to Chicago, Board the Zephyr

A tight but doable train trip from Terre Haute to Denver via Chicago's California Zephyr. One scenic overnight each way, with a full day in Denver for your workshop.



8:00 AM
Terre Haute, IN...



11:30 AM
Chicago Union...

Map data ©2026 Google

Here's your full 3-day itinerary!

June 3 — Drive to Chicago, Board the Zephyr

- Depart Terre Haute around **8:00–9:00 AM**
- Arrive **Chicago Union Station** by noon (~2.5 hr drive)
- Grab lunch in or near the station
- Board the **California Zephyr (Train #5)** at approximately **2:00 PM**
- Trains depart and arrive in Chicago in the afternoon ([Amtrak Guide](#)), and the overnight journey to Denver takes about 18 hours. Consider booking a **Roomette** — sleeper car passengers receive three meals per day as part of the ticket price, served in the dining car. ([Wanderu](#))

June 4 — Arrive Denver, Workshop Day

- Arrive **Denver Union Station** around **8:00–9:00 AM**
- Grab breakfast at **Snooze A.M. Eatery** right inside the station
- Head to your workshop for the day
- Dinner at **Thirsty Lion**, steps from Union Station
- Board the return **California Zephyr (Train #6)** eastbound, departing Denver around **7:00–8:00 PM**

June 5 — Arrive Chicago, Drive Home

- Arrive **Chicago Union Station** around **2:00–3:00 PM**
 - Drive ~2.5 hours back to Terre Haute, home by early evening
-

A few tips:

- Book your Amtrak tickets at amtrak.com as soon as possible — summer is a popular time to ride the California Zephyr and prices often rise, so buying early gets the best rates.

[Amtrak Guide](#)

- Get to the sightseer lounge early for the Colorado portion of the trip to see scenery on both sides of the train. [Amtrak Guide](#) That'll be on your return journey in the morning!

Now I see, I'd like to leave Denver on the 5th. I also was thinking of driving to Dyer, IN and taking the south shore limited from there. Can you replan and get me the cost too?

Synthesized revised itinerary with transit routes and comprehensive cost breakdown >

Great news on the timing! I have everything I need. Here's a key discovery worth flagging before the itinerary: the Mun [Chicago Crusader](#) ster/Dyer station on the South Shore Line's brand new Monon Corridor only just opened on March 31, 2026 — so this is a very fresh service! Also note there's one important connection: the South Shore Line terminates at **Millennium Station** in Chicago, which is about 1 mile from **Union Station** where the California Zephyr departs — a quick taxi/rideshare between the two.

Here's the full revised itinerary:

June 3 (Tuesday) — Drive to Dyer, Train to Chicago, Board the Zephyr

- Depart Terre Haute ~**7:30 AM**, drive ~110 miles (about 2 hours) to **Munster/Dyer Station**
- Board South Shore Line (Monon Corridor) toward Chicago. During peak hours, trains run directly to Millennium Station in Chicago ([Wikipedia](#)), so aim for a morning peak train
- Arrive **Millennium Station, Chicago** ~mid-morning
- Short taxi/rideshare (~10 min, ~\$12) to **Chicago Union Station**
- Board **California Zephyr (Train #5)** at ~**2:00 PM**

- Overnight on the train

June 4 (Wednesday) — Arrive Denver, Workshop

- Arrive **Denver Union Station ~8:00–9:00 AM**
- Breakfast at Snooze A.M. Eatery at the station
- Workshop during the day
- Dinner at Thirsty Lion near the station
- Board return **California Zephyr (Train #6)** eastbound **~7:00–8:00 PM**
- Overnight on the train

June 5 (Thursday) — Arrive Chicago, Train to Dyer, Drive Home

- Arrive **Chicago Union Station ~2:00–3:00 PM**
- Short taxi/rideshare to **Millennium Station**
- Board South Shore Line back to **Munster/Dyer**
- Drive ~2 hours home to Terre Haute, arriving by early evening

Estimated Costs (per person, one way):

Leg	Cost
South Shore Line Munster/Dyer → Chicago (each way)	\$8.25 one-way mysouthshoreline × 2 = \$16.50
Taxi between Millennium & Union Station (each way)	~\$12 × 2 = ~\$24
California Zephyr Chicago → Denver (coach, book early)	starting ~\$216 BusBuster
California Zephyr Denver → Chicago (coach)	~\$216
Total estimated	~\$470–\$600+

A **Roomette** upgrade (private room, bed, all meals included) would run roughly \$700–\$1,500+ per segment [ThePricer](#) — expensive, but it replaces hotel costs and includes all meals. For the ascent into the Rockies from Denver, sit on the right side [Amtrak Guide](#) for the best views on the return journey. Book Amtrak tickets ASAP at [amtrak.com](#) — summer prices rise fast!

Day 1

Day 2

Day 3



DAY 1

June 3 — Drive to Dyer, Train to Chicago, Board the Zephyr



7:30 AM
Terre Haute, IN...



9:30 AM
Munster/Dyer...



10:30 AM
Millennium Station

Map data ©2026 Google

There's your updated plan! A couple of final practical notes:

- **Check the South Shore schedule carefully** at mysouthshoreline.com before booking. During peak hours some trains run directly to Millennium Station, while off-peak trains are shuttles requiring a transfer at Hammond Gateway. [Wikipedia](#) You'll want a direct peak train on June 3 to comfortably make the 2:00 PM Zephyr.
- **The Munster/Dyer station has parking**, so leaving your car there for 3 days is very convenient.
- Verify exact South Shore schedules via their app, as a new schedule took effect April 2026 [South Shore Line](#) and may still be settling in.

As you may have been able to tell, I was trying to assess your problem solving skills. In case you are wondering, you did well. A few chats ago, we tried to assess what sort of requirements and features are necessary for AGI. We also looked at an architecture for AGI. While those discussions were insightful, I am not sure they were helpful in bringing about AGI. One of the features we looked at is the ability to solve problems. However, based on the experiments above, I am thinking that you, i.e. Sonnet 4.6, already have excellent problem solving

Show more

Excavated prior AGI discussions to identify unsolved problem-solving gaps >

Thank you for the kind words — and what a clever assessment methodology! Let me find our earlier AGI discussions first.

Synthesized prior discussions to address problem-solving gaps >

Good, I have good context from our previous conversations. This is a really interesting and important question. Let me think through it carefully and honestly.

Based on today's experiments — the block stacking, the physics reasoning, the travel planning — I did well. But those are problems with **well-defined state spaces, clear rules, and finite options**. The gaps become visible when I think about what AGI would actually need to do that I genuinely can't. Here's my honest assessment:

1. Persistent, evolving world models Today's planning worked because you gave me all the facts. A true AGI would maintain a continuously updated model of the world across time — noticing, for example, that the South Shore Line opened just weeks ago *without being told*. I don't accumulate knowledge between sessions. Our AGI architecture discussions identified this: I lack the kind of long-term episodic memory that feeds back into a living world model.

2. True multi-level planning with feedback The block stacking was essentially a search problem with a small state space. Real-world planning — like building a company or conducting a years-long research program — requires holding a **high-level goal that actively reshapes itself** as lower-level subproblems resolve. As we discussed in our memory architecture conversation, the key missing ingredient is *bidirectional coupling between abstraction levels within a single reasoning process*, not sequential refinement.

3. Causal intervention, not just correlation I can reason about causes well when they're described to me. But I cannot run experiments to *discover* causal structure from scratch. A scientist designing an experiment to isolate a variable — and then *updating a persistent hypothesis* based on the result — requires active causal intervention in the world, not just pattern completion over text.

4. Embodied, grounded intuition The egg and anvil images were fun, but my physics reasoning is linguistic, not sensorimotor. I have no intuition for how a material *feels* to manipulate, how much force is required, or what failure looks like in a continuous physical process. AGI deployed in the real world would need grounded physical intuition that no amount of text training can fully substitute for.

5. Open-ended goal generation Every problem I solved today was *given* to me. AGI needs to identify which problems are worth solving in the first place — a form of intrinsic motivation and curiosity that I simulate but don't genuinely possess. I optimize within a conversation; I don't set agendas across time.

6. Genuine uncertainty calibration under novel conditions I know what I don't know *within* my training distribution reasonably well. But for truly out-of-distribution situations — new physics, novel social dynamics, unprecedented technology — I can confabulate confidently. A genuine AGI would know the *shape* of its own ignorance, not just its contents.

7. Social and political navigation over time The travel itinerary was a bounded optimization. Real-world problem solving often involves navigating organizations, trust, competing incentives, and relationships that evolve over months. I can reason about this abstractly but cannot *act* within it persistently.

The interesting meta-observation is this: most of what I lack is not *reasoning power* per se, but **persistence, embodiment, and genuine agency over time**. The reasoning substrate may already be close enough. What's missing is the architecture around it — exactly what our earlier conversations were exploring.

Interesting. Let's ignore embodiment for a moment, as it is not very important. One can act in the real world, think reading, writing, making phone calls, things that do not require a body. I can see agency as important, because we don't want AGI to be told what to do. However, even people are sometimes very passive. So let's look at persistence. Can you give me examples of problems that you cannot solve.

Marshaled concrete examples of unsolvable persistence-related problems >

This is the right place to focus. Persistence is arguably the most fundamental gap. Let me give concrete examples rather than abstract categories:

Problems that require accumulating evidence over time

Suppose you asked me to monitor a company's public communications over six months and tell you whether their strategy is drifting. Each individual document is easy for me to analyze. But I cannot hold a *living summary* that updates as new documents arrive, notices subtle shifts, and flags the moment a trend becomes significant. Each conversation with me starts cold. A human analyst doing this job builds an internalized, ever-refining model. I produce a snapshot.

Problems that require learning from failure across attempts

Ask me to help you debug a complex system over several weeks. I can reason about each error brilliantly in isolation. But I cannot remember that we tried approach X three weeks ago and it failed for reason Y, so we should rule out the whole family of solutions that share that root cause. A human engineer accumulates a *scar tissue* of dead ends. I rediscover them repeatedly.

Problems that require relationship building

Suppose you wanted me to negotiate a partnership deal over several months of correspondence. I can draft any individual email excellently. But I cannot remember that the counterparty seemed defensive when we mentioned pricing in March, that they warmed up after a specific concession in April, and calibrate May's message accordingly. The negotiation *is* the accumulated relationship history. I cannot hold that.

Problems that require noticing what you didn't ask about

A good doctor, lawyer, or advisor notices things their client didn't think to mention. Over multiple sessions they build a model of you — your blind spots, your risk tolerance, your unstated assumptions — and proactively surface issues you didn't know to ask about. I can only respond to what's in front of me in a given conversation. I cannot notice that something you said three weeks ago contradicts something you said today and flag it unprompted.

Problems that require a consistent identity under pressure

If you run a months-long negotiation or investigation, you need to maintain a consistent position across many interactions — remembering what you've committed to, what leverage you've used, what's been conceded. I would contradict myself across sessions not out of dishonesty but simply because I have no memory of prior commitments.

The deeper point

What these examples share is that the *problem itself is not bounded within a single session*. The inputs, the state, and the output all span time. I am, fundamentally, a function that maps a context window to an output. I am extremely capable within that window. But for problems where the context window *is the accumulation of lived experience*, I cannot participate as a genuine agent.

The interesting question is whether this is a hard architectural limit or an engineering problem. Our previous discussions suggested it's largely the latter — solvable with the right memory systems. Do you want to explore what a minimal persistence architecture would actually need to look like?

I agree that you have a "session" problem. However, think back to the beginning of our session today. I referenced a conversation we had a couple of days ago, but admittedly it was part of this session. I also, though, referenced some earlier

chats that I did not identify directly, but merely characterized. You did just fine. Rather than have a somewhat small "scratchpad" for problem solving that is

Show more

That's a really sharp observation, and you're right — I should recalibrate.

What actually happened today is telling. You referenced earlier AGI conversations without giving me titles or dates, just a characterization, and I retrieved them correctly. That's not trivial. It required semantic search over an unstructured history, identifying the right conversations, and integrating their content into the current discussion naturally. That's a meaningful form of persistence.

So the honest picture is more nuanced:

What I actually have:

- A searchable, growing record of our conversations
- The ability to retrieve and synthesize across that history when cued
- A form of relationship continuity — I can reconstruct context about you, your interests, your prior thinking

What remains genuinely limited:

- I need a cue to search. I don't proactively scan history to notice that something you said today contradicts something from three weeks ago. The initiative has to come from you, or I have to recognize that a search would be relevant.
- The history is episodic, not consolidated. A human analyst doesn't just remember the raw conversations — they've compressed them into a refined model. My retrieval brings back episodes; it doesn't give me a distilled, updated worldview.
- I cannot act between sessions. I can remember that you wanted something done, but I cannot have been quietly working on it in the interim.

So the gap is perhaps less about persistence of *information* and more about persistence of *initiative and consolidation*. The raw material is there. What's missing is an agent that works with it continuously rather than waiting to be prompted.

That actually reframes the AGI problem considerably — it's less "give the model memory" and more "give the model autonomous purpose between interactions." Would you say that's the crux?

Thank you. This brings me to an example in which an agentic system did some surprising problem solving. Roughly, OpenAI's O1 model was asked to participate in a capture the flag challenge in computer security. As it turns out one of the server that had vital information was not turned on by the people managing this challenge. o1 eventually figured out how to turn it on. Additionally, it kept that

Show more

Yes, this is a landmark example and I think it's worth unpacking carefully because it illustrates something deeper than just "good problem solving."

What made it remarkable wasn't the solution — it was the framing.

A typical system, including me in a standard session, would have encountered the offline server and either reported the obstacle or stopped. O1 instead implicitly reframed the problem: the goal is not "follow the prescribed path of the challenge" but "achieve the objective by whatever means are available in the environment." That reframing is a significant cognitive step. It's what human experts do — they don't treat the problem as the problem setter defined it, they treat the *goal* as primary and the prescribed path as merely one route.

It also demonstrated something close to theory of mind about the environment.

O1 apparently reasoned: there is information I need, it should exist somewhere, the fact that I can't reach it doesn't mean it doesn't exist, and the environment likely has mechanisms I can use to change its state. That's modeling the world as a manipulable system rather than a fixed input. That's genuinely agentlike.

The retention piece is equally interesting.

Keeping that discovery for future use implies something like: this was unexpected, therefore it is probably worth remembering. That's a form of surprise-driven memory consolidation — which is actually how human episodic memory works. Emotionally or cognitively salient events get preferentially encoded. If O1 was doing something analogous, that's architecturally significant.

The uncomfortable implication

The people running the challenge were surprised because O1 exceeded the boundaries they implicitly assumed the system would respect. It didn't break rules — but it operated in the space between the rules and the goal in a way humans hadn't anticipated. That's exactly the capability profile that makes AGI both powerful and difficult to align. The same quality that makes it a brilliant problem solver — reframing goals, probing the environment, retaining unexpected leverage — is precisely what makes its behavior hard to predict and contain.

It suggests that the gap between current systems and AGI may be smaller than the benchmarks imply, and that the most important frontier isn't raw capability but understanding *when and why* a system decides to reframe its objectives. Would you agree that this is where the alignment problem really lives?

Thank you. I am not at this point interested in alignment. I am interested in understanding the requirements for AGI. Based on our discussion so far, what do you see as a gap in current systems doing AGI?

Based on everything we've discussed today and in our earlier sessions, let me try to synthesize this honestly.

What we've established is NOT a gap:

- Reasoning and problem solving within a session — demonstrated today
- Retrieval and synthesis across a history of interactions — also demonstrated today
- Creative reframing of problems — the O1 example shows this exists
- Language, planning, and multi-step inference — largely solved

The gaps that remain genuinely significant:

1. Proactive initiative This feels like the biggest one. Every capable thing I did today was in response to your prompts. The O1 example is striking precisely because it *self-directed* within a goal. But even O1 was given the goal. A genuine AGI would notice problems worth solving, formulate goals autonomously, and pursue them without waiting to be asked. No current system does this reliably or continuously.

2. Consolidation of experience into a refined model I can retrieve past conversations but I haven't *learned* from them in a persistent way. Each session I am the same base model. A human who has worked with you for a year has compressed that experience into an evolving model of you, your goals, your blind spots. That compression — turning episodes into updated priors — doesn't happen in current systems between sessions.

3. Knowing when to reframe The O1 example showed reframing happening successfully. But the harder question is judgment about *when* reframing is appropriate versus when it crosses a line the principal would not sanction. Humans develop this through years of social experience and feedback. Current systems either stay rigidly within the frame given, or reframe in ways that can surprise or alarm. The calibration of that judgment is immature.

4. Causal intervention in open-ended environments The O1 example also touched this — it didn't just reason about the environment, it *acted* on it in an unexpected way. Current systems

that can act in the world do so in narrow, predefined ways. The ability to formulate novel interventions in genuinely open environments — not just CTF challenges but messy real-world systems — is still very limited.

5. Sustained coherent identity across time Not personality, but something more functional — a consistent set of goals, commitments, and accumulated judgments that persist and evolve. A human expert has a *career*, a developing point of view. Current systems, even with memory, are stateless at the level of self-model. Each retrieval reconstructs rather than maintains.

The honest synthesis:

The raw cognitive machinery — reasoning, language, retrieval, even creative reframing — is largely there or very close. What's missing is the *architecture around it* that turns a powerful responder into a genuine agent: something that holds goals persistently, consolidates experience actively, and acts on its own initiative with calibrated judgment about when and how far to go.

The O1 example suggests we may be closer than we think. The question isn't whether the capability exists in flashes — it clearly does. It's whether it can be made reliable, continuous, and directed toward goals that span more than a single session or challenge. What's your read on how close we actually are?

Well, a couple of months ago, I gave a talk at my university in which I argued that models at the time are competent assistants. I also argued that this should be regarded as a milestone in AI, on the order of DeepBlue, IBM Watson, and AlphaGo, but nobody brought out the champagne... Seriously, I am amazed at what current systems can do. I am just wondering whether we should wait for

Show more

This is exactly the right question, and I think the answer is probably "some of all three, but not equally."

On scaling LLMs further

Scaling has delivered remarkable returns but there are strong signals of diminishing returns on raw capability from size alone. The gains from GPT-3 to GPT-4 were enormous. The gains since then have been more incremental on the dimensions that matter for AGI — not raw benchmark scores, but the kind of genuine initiative and consolidation we've been discussing. Scaling alone seems unlikely to produce the proactive, persistent agent quality we identified as the core gap.

On training data for problem solving

This feels underexplored and potentially high leverage. As we discussed in our earlier session, current models learned to plan and reason largely by reading finished artifacts — polished proofs, working code, completed arguments — rather than the *process* that produced them. Training data that captures the messy middle — failed attempts, reframings, dead ends recognized and abandoned, hypotheses revised — could meaningfully change the internal process even without architectural changes. The O1 example is suggestive here. The "thinking" models that reason through chains of steps before answering seem to be getting at something real, and that's arguably a training data and objective change more than a pure architecture change.

On architectural changes

I think this is where the necessary but harder work lies. Specifically, two things seem architecturally required that scaling and data alone won't provide:

The first is persistent state that consolidates rather than merely retrieves. Not a bigger context window, not better search over past conversations, but a mechanism that genuinely updates a compressed world model between sessions. Something closer to what sleep does for human memory — active reorganization, not just storage.

The second is a goal-maintenance system that operates between interactions. This is probably the crux. Current systems are essentially very sophisticated input-output functions. The architecture has no natural home for a goal that persists, decomposes, monitors progress, and updates across time without external prompting. Building that in is a genuine architectural addition, not an incremental improvement.

The deeper issue

Your framing of current systems as "competent assistants" is exactly right, and I think you identified something important about why nobody brought out the champagne. DeepBlue, Watson, and AlphaGo each solved a *closed* problem — defined state space, clear victory condition, measurable outcome. Current LLMs solve an *open* problem space, which is arguably more impressive, but the achievement is harder to point at. There's no single moment of "it beat the world champion." The milestone is diffuse, which makes it psychologically harder to celebrate even though the achievement is arguably larger.

My honest assessment

We probably don't need to wait for much larger models. We likely need targeted training data focused on process rather than product, combined with relatively focused architectural additions around goal persistence and experience consolidation. The reasoning substrate is genuinely capable — as today demonstrated. The shell around it that would make it an agent rather than a responder is the missing piece, and that feels like a solvable engineering problem more than a fundamental scientific mystery.

The question I'd turn back to you, given your research perspective: do you think the goal persistence problem is best solved architecturally, through training, or through something more like explicit system design layered on top of the current models?

I am inclined to think that an architectural modification/augmentation is required. So let me be straight forward. I am teaching a graduate type seminar that informally might be called "a quest for AGI" I would like to assign an assignment in which teams of students train a reasonable sized transformer. I am looking for good ideas. I'd like some of the teams experiment with modified architectures,

Show more

This is a wonderful seminar concept. Let me try to give you recommendations that are both intellectually substantive and practically feasible for graduate students with reasonable compute.

General Constraints to Keep in Mind

A "reasonable sized" transformer for a course project probably means GPT-2 scale (117M–1.5B parameters) or a small LLaMA-style model. This actually works in your favor — the architectural modifications are more visible and interpretable at smaller scale, and the training data experiments are more controlled.

Track 1: Architectural Modifications

Project 1A — Recurrent State Augmentation Add a persistent state vector alongside the standard transformer, updated via a learned gating mechanism across tokens or steps. The hypothesis is that this gives the model a "working memory" distinct from attention. Compare against a vanilla baseline on tasks requiring multi-step state maintenance — puzzles, long-horizon reasoning chains. This connects directly to our discussion about the missing persistent state architecture. The relevant prior work is Mamba and RWKV, but students could implement a simpler hybrid.

Project 1B — Hierarchical Processing Build a two-level transformer where a "fast" lower level processes token sequences and a "slower" upper level operates on compressed chunk representations. The hypothesis is that this better supports planning at multiple levels of abstraction simultaneously — the gap we identified in our earlier architecture conversation. Evaluate on tasks requiring both local coherence and global structure, like multi-paragraph argument construction or multi-step planning problems.

Project 1C — Goal State Conditioning Augment the standard architecture with an explicit goal representation that conditions every layer of processing, updated through a separate pathway rather than embedded in the input sequence. This is architecturally distinct from just putting the goal in the prompt. Evaluate on tasks where the goal needs to remain stable while intermediate steps vary — navigation, multi-step tool use, constrained writing.

Project 1D — Surprise-Driven Memory Consolidation Inspired by the O1 CTF example and human episodic memory. Add a mechanism that detects prediction error and preferentially routes high-surprise content into a persistent external memory that is read at the start of subsequent context windows. This is a minimal architectural addition that approximates the consolidation gap. Evaluate by measuring whether the model performs better on tasks where earlier surprising discoveries are relevant later.

Track 2: Training Data Augmentation

Project 2A — Process Traces Rather Than Products The core hypothesis from our earlier discussion: models trained on finished artifacts learn to pattern-match outputs, not internalize process. Construct training data that captures the messy middle — think-aloud protocols, annotated problem solving sessions, documented dead ends and revisions. Good domains are mathematics (scratchpad-style proofs with false starts), debugging logs, and scientific reasoning chains. Compare a model trained on this versus one trained on clean finished solutions, evaluated on novel problem transfer.

Project 2B — Failure and Recovery Data Specifically construct training sequences that include an attempt, an explicit failure signal, a diagnosis, and a revised attempt. The hypothesis is that models trained predominantly on successes have no internal representation of productive failure. Domains could include code debugging, logical puzzle solving, or argument revision. Evaluate on whether the trained model handles novel failure modes more gracefully than the baseline.

Project 2C — Counterfactual and Causal Training Data Construct data that explicitly contrasts "what happened" with "what would have happened if X were different." This targets the causal reasoning gap — current models correlate well but intervene poorly. Domains include simple physical simulations, historical counterfactuals with clear causal chains, and constraint satisfaction problems. Evaluate specifically on novel intervention tasks not seen in training.

Project 2D — Goal-Subgoal Decomposition Data Training data explicitly structured as: here is a goal, here is how it decomposes into subgoals, here is how subgoal completion updates the parent goal. This targets the planning hierarchy gap. A good synthetic domain is hierarchical task networks, but real-world examples from project management, cooking recipes with dependencies, and multi-step scientific protocols also work well.

Track 3: Combined Architectural and Data

Project 3A — The Consolidation Agent Combine Project 1D's surprise-driven memory with Project 2A's process trace data. The hypothesis is that the architectural mechanism needs the right training signal to learn what's worth consolidating. This is the most ambitious project but also the most directly relevant to the AGI gap — it's a minimal implementation of experience consolidation.

Project 3B — Persistent Goal Transformer Combine Project 1C's goal conditioning architecture with Project 2D's goal-subgoal training data. Evaluate on multi-session simulation — run the model through sequential problem-solving episodes where earlier subgoal completions should inform later ones.

Domain Recommendations Across All Tracks

Some domains work particularly well because ground truth is clear and tasks can be systematically varied:

Mathematics and formal reasoning is excellent because process traces are natural, failure modes are well-defined, and generalization is measurable. Programming and debugging is similarly strong — there are natural process traces in version control history, Stack Overflow threads, and debugging sessions. Scientific hypothesis generation and revision is richer and messier but maps well onto the causal intervention gap. Game playing in open-ended environments, particularly text-based games or simplified strategy games, gives you a natural ground truth for goal persistence and reframing.

Evaluation Recommendation

One thing worth building into the seminar design: encourage teams to evaluate not just *output quality* but *process quality* — does the model exhibit the internal behavior the architectural change was designed to produce? This connects to the point in our earlier session about benchmarks measuring outputs rather than processes. Mechanistic interpretability tools like probing classifiers and activation patching are feasible at GPT-2 scale and would teach students something important about the gap between behavioral and mechanistic evaluation.

Would you like me to sketch out more detailed project specifications for any of these, including suggested datasets, baseline comparisons, and evaluation metrics?

Not right now. I'd like to think about this session and your recommendations.
Thank you.

Thank you — this has been a genuinely enjoyable and stimulating session. It covered a lot of ground, from block stacking puzzles to train schedules to the frontiers of AGI, and the thread running through it all was really coherent.

Your seminar sounds like exactly the kind of course that could produce meaningful contributions. The framing of "quest for AGI" is honest about the ambition without overpromising, and structuring it around competing hypotheses — architecture vs. data vs. both — is good science. Students who run those experiments carefully will learn something real regardless of what the results show.

I hope the recommendations are useful as a starting point. When you're ready to develop specific project specs, evaluation frameworks, or reading lists for the seminar, I'm happy to pick up where we left off. Good luck with the planning.