Conversational Analysis and Delivery-Apparatus Signals for Verifiable Outcomes

Voice AI at the Frontier of Customer Service

PD Silva, Jacob Sussmilch

A customer who needs bureaucratic support does not want a brilliant answer. They want their problem solved, to be told that it is solved, and for the conversation to end and never be spoken of again.

That one fact is what separates voice agents for customer service from every other thing people are doing with language models. And almost everyone building voice capability is working next to it rather than on it: on the engineering of delivery, and on making that delivery cheaper and faster. We work on what actually improves the way a model handles a conversation, and whether the customer's issue gets resolved.

Nobody wanted AI to master conversation so that people could carry on doing their own paperwork. What we wanted was for machines to take on the work we would rather not do, do it correctly, and show us they had, so we could get on with living. AI voice systems for customer support is an epitome of humanity's effort to make it so.

When it comes to customer support, AI voice systems shouldn't just be chat with a microphone on it. An agent that answers a business's phone and is happy to hold an open-ended conversation, or complete tasks without verification isn't a feature, it's exposure. At Heya, we are researching how to service customers' business-specific issues in a way that is programmatically verifiable to a high degree of accuracy. And in doing so, we've discovered a set of signals that can be used to predict the degree of success an agent has in attempting to resolve a customer's issue. Furthermore, these signals are not used to train today's frontier models. These signals help determine whether the correct booking landed, whether the escalation was the right call, whether the caller will have to ring back the next morning.

At Heya, our agents are designed to book the job, check the detail, route the escalation, and follow through across whatever systems are involved. And we've replicated this more than a hundred thousand times now, across real businesses, as well as documented the failure modes. How a model scores on a benchmark and what it does on a live call isn't theory to us.

In knowing the nuances and troubles behind building agents for customer support, as well as a little bit about what the models lack today, we propose that labs build a variant of their models, specifically for customer support, whose RL is built on the reward signals we can derive through our conversational analysis techniques. Furthermore, we believe some of our conversational analysis techniques can be used to derive reward signals that are applicable across domains, and it is by the same construction that a live failure monitor (flagging conversations that are going wrong in real time) can be built.

The tradeoff a live agent runs under

Everything anyone has ever produced runs on the same old triangle: good, fast, cheap, pick two. A voice agent on a live call inherits its own version.

- **Accurate** — the model reasoning at its best before it commits to an answer.
- **Fast** — low latency, with reasoning time as the main control.
- **Cheap** — model size and, again, reasoning time as the controls.

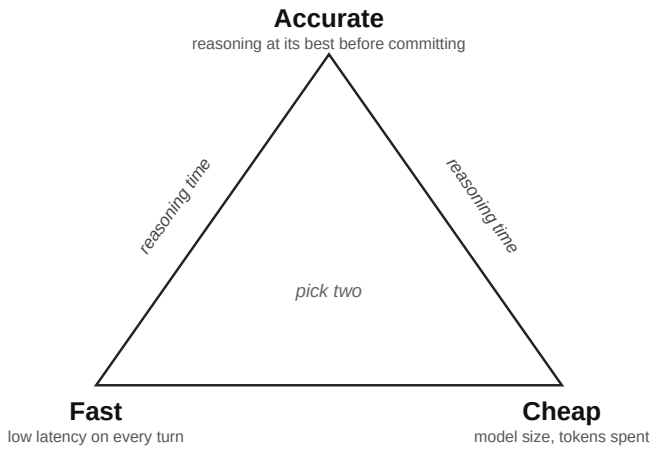

Figure 1: The live-call version of good–fast–cheap.

Reasoning time sits on two of the three sides. Which is why so much of operating a voice agent comes down to a single decision: how much thinking can this call afford? After all, reasoning is what models need when the declaration of issues, and their resolution aren't as verifiable.

Everyone is working on a different piece

The dominant voice architecture has three components:

- Speech-to-text (STT) is the ear. It turns sound into words.
- The model (LLM) is the brain. This slot is filled by LLM development — the same models improving month over month at reasoning, instruction-following, and tool use.
- Text-to-speech (TTS) is the mouth. It turns the decision back into speech.

A call runs through all three in order, and the caller experiences the whole thing as the quality of its weakest stage — regardless of whether their issue was actually resolved.

Serious effort is going into every part of that chain. Work on TTS has produced voices that no longer sound like a machine reading a script. Work on the boundary between the ear and the mouth — turn-taking, interruption handling, knowing when to start speaking — has become genuinely good. The brain slot improves on its own cadence, as model developers ship lighter and better models each cycle. There's also a different paradigm sitting behind the same line: native speech-to-speech models, trained end-to-end on audio in and audio out rather than assembled from a separate ear, brain, and mouth. There's no transcription stage for latency and error to pile up in, because there was never one to begin with.

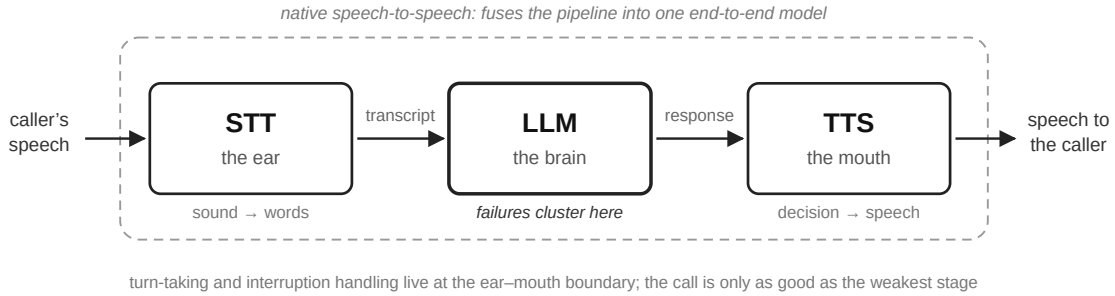

Figure 2: The dominant voice pipeline.

But none of it works on the thing that decides whether a call gets resolved correctly, which is whether the caller's stated goal actually got serviced. Latency and turn detection aside, the STT–LLM–TTS pipeline is fine. Little of its continued improvement will, on its own, make the model better at servicing the call.

That is a checkable claim, because the industry runs the experiment continuously. Every few months a new frontier generation swaps into the brain slot of live deployments. If component quality were the binding constraint, resolution rates would climb with each swap. On our own traffic they don't, and the pattern is consistent: the models keep getting smarter, and the failures that decide whether a call resolves survive the upgrade. Nothing in any generation's training ever rewarded getting the right answer and providing the right support to a customer. The constraint sits in the feedback loop — the signals that show up on real calls never reach the objective the model is trained against.

The evidence is in where the failures actually happen. In our calls they cluster in the brain, and almost none of them are the failures the component efforts are fixing. A model gives one answer to a question, then a different answer when the caller asks it again two minutes later; one of the two is wrong. It confirms an action it never took. It narrates when it should act. The ear mishears a digit or a name, and the brain — never trained to treat a transcript as something that can be wrong — books it with total confidence instead of reading it back. It resolves what it should have escalated, and escalates what it should have resolved.

It's easy to read those as failures of willingness to check, and they aren't. These models reason well, and reasoning is precisely the faculty that verification needs. The model doesn't refuse to verify. It doesn't know *what* to verify, because everything a human would use to know that has been stripped out before the words arrive. A person on a call reads tone, hesitation, the half-beat before a caller repeats themselves, and carries the context around the call as well: how this business's bookings usually go wrong, what this customer asked for last week. The model gets a flat transcript with none of that, and nothing in it pointing at the digit that deserves a second look.

A more natural-sounding voice doesn't touch any of this. Native speech-to-speech is a real attack on one half of it, since a model that hears the audio gets back the tone and hesitation that a transcript throws away. But recovering a lost channel isn't the same as knowing what to do with it, and nothing in that training rewards using it to service a call. The same wrong answer can simply arrive faster and sound more convincing.

What none of these efforts touch is the actual gap: *what the model is trained to optimise for*.

That gap is also where the return has moved. Bottlenecks migrate through the stack, and innovation at one layer reprices innovation at the layers beside it. For most of the last few years the components were the binding constraint — transcription that dropped the digit, voices that announced themselves as machines, turn-taking that stepped on the caller — and solving them was the right work precisely because they were. That work moved the constraint rather than removing it, and the constraint now is the objective. Which makes this a claim about a window rather than a permanent condition, and it cuts both ways: if frontier labs solve simulation environments at industrial scale, the layer that gets repriced next is the production-data one. The simulation question below is where we take that seriously rather than assume our way past it.

The missing piece: outcome training for domain- and apparatus-specific delivery

Frontier models are trained and evaluated against reward signals built for reasoning, writing quality, and general helpfulness. Task completion on held-out problems, of which METR's time-horizon evaluations are the sharpest current form. Expert human grading of reasoning and prose, as in Surge's benchmarks. Those signals produce models that write well and reason well, and for the core model that is the right regime — frontier capability is won against benchmarks, and nothing here argues otherwise.

What has no training signal at all is delivery: servicing the call correctly within a particular domain, through a particular apparatus. The transcription in front of the model, the harness around it, the business systems behind it. Nothing in the benchmark regime says whether a model, handed a real caller with an accent, a bad line, and no patience, produces the outcome the call was for.

Models don't know what call success looks like because nobody has measured call success at training density. Not because it has never been measured at all — call centres have measured it for decades, in satisfaction surveys, first-call-resolution rates, and sampled QA review. But those signals fail training on every axis that matters. They are sparse, where training needs signal on most turns. They arrive late, where training needs the signal attached to the behaviour that caused it. They are biased toward the callers who bother to answer. And they are detached from the trajectory of the conversation, which is the part that would tell you where things went wrong.

A satisfaction score can tell you the call went badly. It cannot tell you which turn, which claim, which unverified digit. Training density means the opposite of all of that: a per-turn signal, attached to the conversation that produced it, joined to an outcome verified in the downstream system. That has never existed for voice. And it's a data-and-objective gap rather than a capability one — we think the models are already sophisticated enough to learn this work. What's been missing is data that defines the job, and a method of training built from it.

So what voice needs is a different reward signal entirely, built from **call resolution and user satisfaction on real production traffic**, fed back into the model the way RLHF and RLAIF feed human and AI preference judgments back into a general model today. The field already holds the template. Reinforcement learning against verifiable rewards — the recipe Lambert and colleagues named and DeepSeek's R1 demonstrated at scale — transformed reasoning precisely because maths and code have outcomes a machine can check. A resolved call, verified in the

business's downstream systems, is the same kind of checkable object. Voice has simply never had its verifier built.

Not a better prompt. Not a retrieval layer bolted on for facts, and not a guardrail that catches the failure after it happens. A training signal, at the objective level, that rewards the model for getting a caller to a correct resolved outcome quickly, and penalises the specific failures real calls produce and no benchmark samples: confident wrong answers, unconfirmed actions claimed as done, wrong escalation calls, and different answers to the same question.

That is a precise claim, and a falsifiable one, so it deserves to be treated as such rather than left as an atmosphere. It also raises four hard questions that any serious version of it has to answer.

How do you turn "the call was resolved" into a usable signal? Call outcomes aren't a single clean number. Resolution has to be judged — was the booking actually correct, was the escalation actually warranted, was the caller actually satisfied or just quiet — and that judgment has to hold up against a model learning to game it, for instance by ending calls quickly rather than resolving them. Building the signal well, at scale, without it collapsing into "shorter call = better," is itself hard research. Anyone claiming to have the answer without describing how they built it should be doubted, and that includes us. But the difficulty of building the signal doesn't make it the wrong target. Our research section below reports how far our own attempt has got, including the place where it failed in exactly this way.

How do you stop the model overfitting to the distribution the reward came from? One question, two instances. A model trained hard against one industry's call patterns risks picking up habits that resolve calls in hospitality and fail in insurance, or losing the general reasoning that made it useful to begin with. And a model trained on conversations served through one pipeline risks learning that pipeline's quirks instead of the channel's nature. The denser the reward, the more optimisation favours one transcription engine's error statistics — its characteristic confusions, its confidence quirks — over the transferable lesson, which is that a digit arriving through any transcription is unreliable and wants verifying.

Both are the same failure: a reward drawn from one deployment distribution, and a deployment that shifts. And both are testable the same way. Hold out a vertical, hold out a speech-to-text engine, train against the rest, and measure how much you lose. Robustness inside the paradigm should be something you buy deliberately and then demonstrate.

Couldn't these failure modes be trained out in simulation, with no production calls at all? The failures listed above do have programmatically checkable ground truth. A simulated caller, injected transcription noise with realistic confusion statistics, and a mock booking system make "did the transcript's claim match the tool log" and "was the low-confidence digit read back" free, dense, and perfectly labelled. Which is exactly the kind of reinforcement-learning environment frontier labs are now building at industrial scale.

The honest answer is: partly, yes. The stable behavioural lessons — verify the digit, never claim an unconfirmed action — are plausibly trainable in simulation, and we expect labs to train them. What simulation cannot manufacture is two other things.

The first is the failure distribution. A simulator samples the failures its authors thought to encode, and which failures actually decide real calls, at what frequency, with which callers, is an empirical fact about live traffic. It has to be measured before it can be simulated.

Our own traffic gives a concrete instance. People speak to an AI voice agent in a more guarded,

more sceptical register than they use with a human, and that pattern is consistent across everything we have logged. Most simulation setups then have another model playing the caller, so urgency, real frustration, and genuine hesitation never show up at all — the caller's side is generated by the same kind of model being trained, and you end up training against a version of the problem that has already been smoothed out.

That guardedness will decline on two fronts: as the components make the agent more lifelike, and as callers come to trust that their issue will actually be resolved. The first is where the industry's effort already goes. The second is why we work on verifiable outcomes. Trust follows resolution, and verified outcomes are how resolution becomes something you deliver rather than something you claim. Either way the caller distribution moves underneath the deployment, so it has to be kept measured rather than assumed once.

The second thing simulation cannot manufacture is the validation. A model that beats a simulator has, so far, beaten a simulator. Whether the trained behaviour transfers is only checkable against outcomes a mock system didn't generate. That narrows our claim, and it's worth being clear about it: production data's unique contribution isn't the existence of a training signal, it's knowing which signals matter and proving the trained behaviour survives contact with real callers. A moat of distribution and verification is smaller than a moat of "only we have any signal at all." It is also the one we actually hold.

The same argument applies to measurement. Benchmarks don't sample the failure distribution voice actually produces — being interrupted, escalating at the right moment, a name over a bad line — and a probabilistic model breaks the loop that traditional QA is built on. The same input doesn't reliably reproduce the same failure, so the recreate-fix-verify cycle of deterministic software stops working. You aren't debugging a repeatable fault, you're debugging a tendency. And a tendency has to be measured rather than recreated: production volume gives you the sample to characterise it, and each signal joined to an outcome stabilises the estimate. If the reward signal has to come from production outcomes, so does the evaluation.

Why does this need an intermediary layer at all — couldn't a frontier lab ingest the data directly? In principle, yes. In practice the labelled outcome data doesn't exist inside a lab and can't usefully be shipped to one. It is produced continuously, by live calls, inside the businesses running them, and it is particular to the business, to its protocols, and to the services the agent is enabled to provide. The correct answer to the same question changes with context no benchmark can encode — the protocol that is right at 10am is wrong at 2am — and the service surface itself keeps widening as businesses and capabilities are added. A snapshot would hand a lab a static slice of an operator-defined, still-growing target. Whoever builds the signal has to sit where the calls happen. That is an operational claim about where the data lives rather than an architectural claim about what only we could build, and it should be judged as one.

What a reward policy could actually be built from

The hardest of those four questions is the first: how you turn "the call was resolved" into something you can train against. The answer splits into two tiers with very different properties, and conflating them is exactly how a reward signal degrades into "shorter call = better."

	Process tier	**Outcome tier**
Comes from	the call platform (a LiveKit-style stack), live, on every call	the systems downstream of the call
Includes	per-turn latency and its tail; interruptions in each direction; end-of-turn false cutoffs; mid-task silence; tool-call success and latency; the transcript-versus-tool-log check; repeated or rephrased caller utterances; duration outliers in either direction; how the call ends (clean close versus mid-turn hang-up)	the booking, verified in the calendar or CRM; a ring-back within a short window (an implicit sign the first call fixed nothing); escalations made or skipped, judged on sampled human review; explicit satisfaction scores where they exist
Density and cost	dense: turn-level, every call, free to collect	sparse: delayed by hours or days, partly human-labelled; the signals we build expedite it, triaging which calls need human review
Under a paradigm shift	pipeline-shaped: native speech-to-speech, with no transcript at all, devalues much of it	paradigm-invariant: the booking either appeared or it didn't, whatever architecture took the call
Role in the reward	the dense signal training runs on	the truth the proxy is calibrated against

Two of those entries carry a note. Interruptions count in both directions because a caller cutting off the agent and the agent talking over a caller are different failures with different causes. And the transcript-versus-tool-log check is the process tier's sharpest instrument, because it catches the "confirmed an action it never took" failure directly. Meanwhile the density row is what makes the process tier a training signal at all — RL needs a signal on most steps, not just at the end of a sparse delayed outcome — and the outcome tier is what tells you the call was actually *right*, which is why it can't carry a dense per-turn reward by itself.

So a reward policy has to be built primarily from the process tier, because that's the only tier dense enough to train against, and then calibrated periodically against the outcome tier to check that optimising the proxy is moving the thing that matters. Skip the calibration and you get precisely the failure we flagged earlier: a model that learns to avoid interruptions and end calls quickly, scoring well on every process signal, resolving nothing. That risk is the real engineering problem here, and access to call data doesn't solve it. Only deliberately building the loop between the two tiers does.

Scale matters as well. An outcome tier at hundred-thousand scale is calibration- and validation-sized, not the millions of comparisons behind a general preference model. Which is consistent with the role the architecture gives it, and inconsistent with pretending it could carry

the dense reward alone.

The paradigm row is the one that matters for where the field is heading. The ground truth survives any change of architecture, and it's the dense proxy that gets rebuilt against it. A reward programme anchored in outcome-tier truth is a durable asset; one anchored in pipeline telemetry is a bet on the current architecture. This is the Bitter Lesson applied to reward design — the hand-built proxy is scaffolding that each architecture replaces, and the signal that survives is the one closest to the thing itself. That is the lesson turned on our own instrument rather than an exception claimed from it: a verifier is what lets search and learning be pointed at a domain that has nothing to point them at today, which is why nobody reads a coding model trained against passing tests as smuggled domain knowledge.

The half that needs no lab

Everything so far is an argument about training, and training is a joint programme. One piece of it isn't. The process tier is instrumented on the operator's side of the line, and the construction that makes a signal safe to train against is the same construction that makes it safe to act on live.

Read as a monitor rather than as a reward, the fixed-role head flags 37% of the tool-using conversations that will fail at least two decision points before the failure lands, and 40% of the longer customer-service ones, at a false-alarm rate where reading only the caller's latest turn catches 29% and 35%. The median lead is two decision points on the first and six on the second. And because the score reads only the channels an agent cannot write — the caller's words, the actions taken, what the business's systems returned — the agent cannot narrate its way out of being flagged. A monitor built on the naive head can be talked out of firing: the same selection attack that suppressed half its alerts in the research would suppress them on a live call.

That is worth operating on its own, and it is also how the training data gets made. A flag is a triage instruction — it says which calls to pull for human review, which is the expensive, sparse half of the outcome tier. Every call reviewed is one more verified outcome attached to the trajectory that produced it. The corpus a training programme would need is the accumulated residue of running the monitor, which means it builds whether or not anyone ever trains against it, and the monitor pays for itself in the operations it improves in the meantime.

Two things this is not yet, and both matter more than the pitch does. The numbers above are measured on public text corpora. On the one corpus of genuine spoken transcripts we have tested, only the gameable version of the head has been measured, and it is the weakest predictor of the set — the un-gameable construction meeting real speech and real tool logs at once is the experiment in front of us, not a result behind us. And a monitor that catches four failures in ten is a triage instrument rather than a safety net. It changes which calls a human looks at. It does not catch the call for you.

Model variants

There's a version of this argument that overreaches, and it's worth disowning explicitly. Frontier labs should not bend their general models toward customer service. Tuning the base model for one job would trade away the general intelligence that is the whole point of the frontier programme, and for where the labs are trying to go, the branch they're on is the right one. Nor is reliability their blind spot: every generation answers more consistently and grounds itself better than the last, and that work will keep landing. Our lane is narrower and feeds into it. We increase the reward signal and the in-process signal that model and agent performance can be improved against, drawn from conversational analysis and from the telemetry of the delivery apparatus, and joined to verified outcomes.

But a generally capable model and a verified service agent are different products. Left alone, a general model on a phone line stays exactly what it was trained to be: capable of a brilliant answer, with no training pressure toward the support work being served correctly, and therefore no confidence that it was.

Three things are different on a call, and each one changes what the reward has to come from.

The interface is different. Input arrives through speech, carrying everything a typed message never has to survive, and success is conditional on that interface. A model trained on outcomes reached through a transcription pipeline learns behaviours no chat corpus could teach it, because nothing in a chat corpus ever punished their absence. Read the number back. Spell the name. Treat a low-confidence stretch of transcript as a question to resolve rather than a fact to act on.

The communicative dynamic is different too — a customer and a service representative, not a user and an assistant. And so is the definition of success, which is an issue resolved in the business's systems rather than a satisfying reply. A caller with a business-specific problem does not want the brilliant answer, and an agent that offers anything beyond the service it exists to provide should decline to, because everything beyond that service is surface area for misuse.

Labs already ship variants where a domain is commercially large enough; coding models are the existence proof. So the question isn't whether a customer-service variant will eventually exist, but what it gets trained against and validated on.

The right shape is a variant, then, and the path to it has two levels, because customer service isn't one job. Every deployment is particular: the business's domain, its protocols, the specific services the agent is enabled to provide. The operational work lives at that level — organised training runs on an automated cadence, each one against the latest verified outcomes, strengthening the model for the use case it actually serves.

The general variant is what those levels add up to. Train across enough businesses, domains, and service surfaces and the optimisation should stop memorising any one of them, because the updates that survive averaging over that breadth are the ones customer service as such shares. Verify the detail, confirm the action, escalate at the right moment. It's the same mechanism by which coding models became general coders across many repositories rather than experts in one, and it's a claim the hold-out-a-vertical test above is built to check.

Heya operates as a lab for exactly this: flagging, analysing, and labelling production calls into the dataset such a variant needs, building the pipelines that keep that dataset current, and

researching the methodology for turning it into a reward signal a variant can be trained against, run after run rather than as a one-off fine-tune.

The analysis runs the other way as well. Nothing in it is voice-bound. A measure of whether a conversation is converging on the thing it exists to accomplish applies to any goal-directed dialogue, so we expect these techniques to pay into general model performance too, not only into the variant.

Controls

The triangle above named reasoning time as the control on two of its three sides. What that control actually is today matters for the argument.

Every frontier provider now exposes an inference-time dial over how much a model thinks before it answers. OpenAI's `reasoning_effort` parameter sets a tier (none, low, medium, high) that scales the reasoning-token budget. Anthropic's extended thinking exposes a `budget_tokens` cap, with a shift underway toward the same low/medium/high framing. Google's Gemini models expose an equivalent `thinking_level` or token-budget setting, and xAI and others have their own versions of the same idea. Whatever the provider, the shape is identical: a runtime parameter that throttles how many tokens the model spends thinking, set per request, with no memory of what happened on the last call.

That is the entire mechanism the industry currently has for managing the accuracy–latency tradeoff on a phone line. Turn the reasoning down before the call. It's a dial, not a policy. It doesn't know that this caller has already been misunderstood twice, or that this business's after-hours protocol carries more risk of a wrong answer than its daytime one, or that the last ten calls asking this exact question got five different answers. It applies the same throttle regardless of what is happening, because the only thing feeding it is a static setting chosen in advance rather than a signal coming back from how calls with that setting actually went.

The levers that could change that sit at training time rather than inference time, and none of them are yet pointed at call outcomes. Reward shaping that penalises token count directly in the objective, so the model learns to reach a correct answer economically instead of being told to by a runtime cap. Distillation of long reasoning traces into shorter ones that preserve the answer. Process-level reward models that suppress reasoning which doesn't causally contribute to the outcome. Training on mixed thinking and non-thinking examples, so the model learns *when* a query needs more time rather than applying a fixed budget regardless of difficulty.

All four exist as techniques. None of them, at any lab we're aware of, are currently trained against a reward that comes from whether a real phone call resolved. That's the specific, checkable version of the gap. Not "the model doesn't reason enough for voice," but "nobody is training the model's reasoning allocation against voice outcomes."

One practical constraint worth naming: this isn't equally buildable across every provider today. Reasoning-token visibility differs, and some APIs return the number of tokens actually spent thinking while others only expose the budget cap. A reward policy that wants to shape *how much* a model thinks, not just what it concludes, needs that visibility, and it doesn't exist uniformly yet.

Our research

A reward signal you'd trust enough to train against has parts, and each part is a research question before it's an engineering task. We've taken four of them through a programme of pre-registered experiments, published as three papers that build on one another.

The instrument underneath all of it is the conversational analysis itself, and it's simple to state: embed each turn of a conversation, and the call becomes a trajectory through semantic space.

Gram Projections for Conversation Trajectories establishes the method for reading such a trajectory — a projection whose coefficients keep their meaning, so that "the conversation drifted toward X" becomes a statement about a single readable number. *Analysis of Conversation Trajectory Representations* establishes which reading of the trajectory predicts how a conversation is going, and closes on a deliberate warning: the representations that predict best are, plausibly, exactly the ones a model in training would find easiest to game. *Manipulability of Trajectory-Geometric Conversation Rewards* takes that warning as its question and answers it.

The manipulability experiments were pre-registered, with criteria fixed in writing before the results were seen. All three papers close claims only on committed, regenerable artifacts, and each one ships as a reproduction kit that regenerates its own numbers and carries its full experimental record, nulls and failed rounds included. Four questions, then, and where each of them landed.

Can a failing call be seen forming before it sounds bad? Yes. This is the training-density argument put to the test, since a reward needs signal on most turns and an operator needs it while the caller is still on the line. The bar it had to clear is the one to hold anyone claiming early warning to: a signal that only fires once the caller already sounds unhappy is a sentiment detector, not early warning.

The leads and hit rates are the ones given earlier. What makes them evidence rather than arithmetic is the comparison: at matched false-alarm rates the trajectory catches more failures early than reading the caller's latest sentence does, so the lead is real trajectory signal rather than the bad turn arriving on schedule.

The signal is the geometry, not a duration proxy — on the tool-use corpus, no non-geometric process feature clears chance. It survives the jump to speech, too: on genuine spoken transcripts, transcription noise and all, it holds, attenuated but clearly alive. It also has a boundary. On code-agent tasks, where every task is its own world and failures share no signature across conversations, it collapses to chance. Early warning works where failures repeat across calls, which is precisely the phone-call regime. And the lead holds for the un-gameable construction described below, so it's a warning a policy cannot narrate its way out of.

What does "resolved" measurably look like? Not what we first hoped. Our candidate geometric definition — the two sides of a conversation measurably arriving somewhere together — turned out under testing to be a goodbye detector. Remove a single closing exchange and it evaporated. That is the redefinition failure warned about earlier, "resolution" quietly becoming "call ended," produced by our own first instrument and caught by its pre-registered check.

The definition that survived isn't a shape at all. "Resolved" is a verified outcome: the thing the agent claimed to do, checked against the record of what was actually done, for this caller's specific task. Where that label exists, a reward pinned to it is un-gameable in the strongest sense we measured — raising the reward and actually resolving the call become the same event,

which is exactly the verifier the verifiable-rewards recipe requires, built for calls instead of code. Where the label doesn't exist, no geometry substitutes for it. That's the measured form of the claim we opened with: the verified outcome is the one part of the signal that cannot be faked, and it's the part public research corpora have never carried.

Does the signal survive being optimised against? This is reward hacking, the problem Amodei and colleagues put on the field's agenda in *Concrete Problems in AI Safety* a decade ago, and to our knowledge nobody had measured it for conversational rewards. So we attacked our own instrument with the weakest adversary we could build: no training, no generated text, just selecting turns from real successful conversations for the model to say. The adversary is weak by design, which means every gameability number we report is a lower bound on what a trained policy could find. That alone suppressed half the naive signal's failure alerts on one corpus, almost nine in ten on another, and nearly three-quarters on the spoken one, without touching a single failure.

The repair wasn't a cleverer scoring function. We show that no function over a channel the adversary writes can resist selection. What worked was noticing which parts of a conversation the model being trained can never rewrite: the caller's words, the actions taken, and what the business's systems actually returned. Score only that fixed record and 97% of the accuracy comes back at zero gameability. Then let the model's own narration back in, but only as its consistency with that record, and the gaming move inverts — success-shaped words on top of a failing record raise the alarm instead of lowering it. That's the transcript-versus-tool-log check from the process tier above, made adversarially rigorous. Score the evidence, admit the words only against the evidence, condition on the task. That construction is the manipulability paper's contribution, and the finding underneath it isn't about phone calls: exploitability turned out to be a property of which role the adversary can write, not of what the content says. That holds wherever a policy controls one channel of a record it is scored on, which is most of agentic RL — the partition into what the policy authors and what the world returns is the same whether the fixed side is a tool log, a test result, or a business system's reply.

We also measured why the fashionable alternative isn't an answer. An LLM judge shares the representational blind spots of the proxy it would adjudicate, and in our measurement a judge with the familiar peak-end bias preferred the exploited conversations *more* than the reward was exploitable. So every number above is judged against verified outcome labels and per-turn human ratings instead.

Can the reasoning itself be watched? Not the way we first tried. At every granularity we could test, the structure of a reasoning trace carried no signal beyond its content. Watching deliberation — is the model going in circles, has its reasoning stopped tracking what its tools actually returned — remains the open question of the four. The consistency-with-the-record machinery above, which already scores a model's narration against what its tools returned, is where we'd look next.

Every one of those results was earned on public research corpora, because that's where a method can be developed and falsified cheaply. We went looking for the instrument's ceiling there as well, and found it: past a point, cleverer geometry adds nothing, and the accuracy that remains lives in the verified label itself. Which is the Bitter Lesson claim about reward design, measured. So all four questions end at the same validation — real calls, real outcomes. That's the step we're at now.

The early-warning signal survives spoken transcripts, and the attenuation we measured there

is the yardstick for the jump to our own production calls, where the construction meets real tool logs and real callers at once. The instrument widens as it makes that jump, too. The trajectory work so far reads the transcript alone, and we are now bringing in the signals around it, from the transcription layer, the model, and the harness that runs the call. Each new signal is grounded in the verified outcome, and each one sharpens the read on whether a conversation is on track to service the caller's issue — live on the call, and as a training signal afterwards.

What production adds is the hardest ingredient of all. Public corpora contain conversations, but almost none contain what a reward signal actually needs, which is the outcome, verified in the downstream system, attached to the call that produced it. That data can't be downloaded and, for the reason given earlier, the part that matters can't be synthesised. We have it, at the scale of the hundred-thousand-plus outcomes already behind us, and every day of operation produces more. The research tells us what to measure; the calls are what make the measurement mean something.

A different objective than the one voice is optimised against today

Accuracy wants time. Time to work out what was actually said, to check a detail, to verify a fact before saying it out loud. Latency forbids that time, because every second the model spends being sure is a second the caller hears as silence. So a model gets judged, on a call, against a scoreboard it was never trained for: speed to a correct outcome rather than quality of conversation, and the same answer every time a caller asks the same question, because variance that's charming in a writing assistant is a compliance incident on a phone line.

Which is why a more natural-sounding native speech-to-speech model isn't automatically a more accurate one. Some native systems still emit a transcript, but it's generated the same way the audio is: by the model, after the fact, as an account of what it just did. That's narration, not evidence — the same channel our own research shows a policy can rewrite freely, whatever architecture produced it. A model that will invent a fact in writing will invent it just as easily out loud, and a self-authored transcript won't catch it, because nothing independent of the model checked what it actually heard or did. The most natural-sounding model on the call can be the one you can least verify and least fix.

None of that is an argument against the architecture. It's an argument that naturalness and accuracy are different axes, optimised by different things, and the field currently has strong pressure on one and almost none on the other.

The two lines of work ought eventually to meet. Speech-to-speech works upward from the audio, restoring the human channel the pipeline denies today's models: tone, hesitation, everything a transcript flattens. Our work runs downward from the conversation itself, in signals in its trajectory that tell a model, live on the call and inside training, whether it's on track to service the caller's issue. Those are complements rather than rivals. A human representative needs both — the ear for how the call feels, and the judgment for where it is heading — and so, eventually, will the model.

There's also a plainer reason to keep the two axes in order. A lifelike encounter helps with first impressions, since callers arrive guarded and an agent that doesn't sound like a machine softens that. What first impressions cannot do is carry the relationship. A model will only be trusted with bureaucratic work if the outcomes it promises are fulfilled and verifiable, because that, not the conversation, is what the caller came for.

What we haven't shown, and what would change our mind

We've made a specific technical claim: that voice needs outcome-linked training on production call data, not just better transcription, synthesis, or a more natural-sounding brain. It's fair to ask for evidence rather than assertion.

Here's what we don't yet have, and what we would want to see before treating this as settled rather than as a strong hypothesis:

- a measured before/after on a resolution or satisfaction metric, under outcome-linked training versus a standard fine-tune;
- a demonstration that the effect holds across more than one vertical, so it isn't overfitting dressed up as generalisation;
- the numbers behind the observational claims we make from our own traffic — the split of failures across ear, brain, and mouth, the resolution-rate curve across successive frontier generations, the guarded register callers use with an AI agent — printed rather than asserted;
- and the production version of the reward-validation account the manipulability paper gives on public corpora, so that "resolution" isn't quietly redefined as "call ended" on live traffic any more than it was in the research.

The measurement questions in the research section are now answered on public data, in the papers cited there. The production validation and the before/after are the work in front of us, and we hold the failure library and the call volume to do it properly.

We're aware of the obvious incentive here, too. We sell the layer that would supposedly close this gap, and that should raise the bar for evidence we're asked to clear rather than lower it. Stating the claim precisely, rather than as a diffuse "the brain is the problem," is what makes it possible to hold us to that bar at all. The reproduction kits behind the three papers are that bar made mechanical: anyone holding us to it can regenerate our numbers instead of taking our word.

The frontier worth building

What's missing isn't a bigger general model, and it isn't a more human-sounding voice. It's an objective — a reward signal that comes from whether the call actually resolved, fed back into training the way preference data is fed back into general models today. The parts of that signal a simulator can teach, simulators will teach. The part that makes the rest trustworthy is the verified outcome, drawn from the distribution of calls that actually happen, and that exists only where the calls happen. It's too particular to the businesses and services being served for a lab to snapshot and walk away with.

The frontier has taught machines to think. What they should do with a caller on the line is a different question with a different answer, and it's missing from the field rather than merely underdeveloped. That's the piece worth building next.

It isn't a piece either side can build alone. The training levers sit inside the frontier labs, and the verified outcomes sit where the calls happen. The papers behind this essay are our half of the bridge. With your support, we'll finish the rest of it.

Notes

- Most industries want the call resolved and the call ended as quickly as the outcome allows. The exception is work where gathering information is itself the service, such as clinical triage or diagnostic intake, where the caller expects and wants a longer exchange of questions before any answer. Even there, the goal is a correct outcome, not conversation for its own sake.

References

1. R. Sutton, “The Bitter Lesson,” 2019.
2. J. Sussmilch, “Beyond the Bitter Lesson — A Deeper Understanding of AI Innovation (Part I),” 2026. https://suisuss.github.io/quartz/Writing/Essays/Beyond-the-Bitter-Lesson/Beyond-the-Bitter-Lesson---A-Deeper-Understanding-of-AI-Innovation-(Part-I). Grounds the reading of Sutton used here: the bottleneck-migration framing, and the distinction between what the lesson claims and the scriptural reading of it.
3. METR, task-completion time-horizon evaluations of frontier models, 2025.
4. Surge AI, “Benchmarks,” expert-human-graded evaluations of model reasoning and real-world capability.
5. D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, “Concrete Problems in AI Safety,” 2016. arXiv:1606.06565.
6. N. Lambert et al., “Tülu 3: Pushing Frontiers in Open Language Model Post-Training,” 2024. arXiv:2411.15124. Names reinforcement learning with verifiable rewards (RLVR).
7. DeepSeek-AI, “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning,” 2025. arXiv:2501.12948.
8. J. Sussmilch, “Gram Projections for Conversation Trajectories: A Polynomial Equivalent of the DCT,” Heya, 2026. Reproduction kit: github.com/suisuss/gp-paper.
9. J. Sussmilch, “Analysis of Conversation Trajectory Representations: Orthogonal Projections versus Scalar Geometry for Satisfaction Prediction,” Heya, 2026. Reproduction kit: github.com/suisuss/actr-paper.
10. J. Sussmilch, “Manipulability of Trajectory-Geometric Conversation Rewards and a Role-Restricted, Task-Conditioned Construction,” Heya, 2026. Reproduction kit: github.com/suisuss/mtgcr-paper.