ARC-AGI-3 • PUBLIC RESULTS

Two 100%s on ARC-AGI-3

with zero model calls

We've been playing ARC-AGI-3, the interactive reasoning benchmark from ARC Prize. **Unlike earlier ARC tasks, these are games:** a 64×64 grid of coloured cells, a handful of buttons, no instructions, and no explanation of what any of it means. You work out the rules by playing.

> **Headline result**
> ft09: 6/6 levels, 80 actions, 100.0%. tr87: 6/6 levels, 322 actions, 100.0%. No LLM is used for perception, planning, or action selection.

Official scorecards

Every result below is an official ARC Prize scorecard with a replay.

Game	Levels	Actions	Official score
ft09	**6 of 6**	**80**	**100.0%**
tr87	**6 of 6**	**322**	**100.0%**
cd82	2 of 6	21	8.59%
bp35	2 of 9	93	6.67%
lf52	2 of 10	42	5.45%

ft09 Open official scorecard →
tr87 Open official scorecard →
cd82 Open official scorecard →
bp35 Open official scorecard →
lf52 Open official scorecard →

The number that matters

ARC-AGI-3 doesn't score you on finishing. It scores **Relative Human Action Efficiency**: your actions against a baseline set by first-time human players, squared, weighted so later levels count for more. Taking twice as many actions as a human doesn't score half. It scores a quarter.

> **ft09 efficiency**
> Human baseline: 208 actions across six levels. Our run: 80 actions. Every level hit the benchmark's 115-point per-level cap.

Level	Our actions	Human baseline	Level score
1	4	43	**115**
2	7	12	**115**
3	14	23	**115**
4	16	28	**115**
5	26	65	**115**
6	13	37	**115**

Several other levels cap too: cd82 level 1 in 7 actions against a human's 55; lf52 level 1 in 8 against 32; tr87 level 1 in 28 against 54.

tr87 finished 6 of 6

tr87 went from four levels to all six. These are the scorecard's own per-level figures:

Level	Our actions	Human baseline	Level score
1	28	54	**1.150 (capped)**
2	30	58	**1.150 (capped)**
3	39	40	1.052
4	29	45	**1.150 (capped)**

5	50	71	**1.150 (capped)**
6	146	146	1.000

Six of six, 322 actions, no resets, against a human total of 414. Levels 5 and 6 are the ones nobody had opened.

Level 5 inverts the puzzle. Both words are fixed and the legend itself is what you edit: you are handed a translation and asked which rules produce it. Dialling it out isn't an option; eight panels of seven glyphs is 5,764,801 states. What makes it finite is that the rules have to tile both words exactly, leaving 24 orderings, and most of those ask a panel for a glyph it can't display.

Level 6 combines every earlier mechanic. One glyph rewriting as several, longest-first segmentation, an editable legend, rotated glyphs, and a chain through a colour that appears in neither word. The last level of one of these games seems to be an exam on the rest of it.

Zero model calls

> **No LLM in the loop**
> Not for perception. Not for planning. Not for choosing a move. The agent reads the raw grid, decides, and acts. Total inference spend across every run above: $0.00.

That isn't a cost story, it's a capability one. Under a squared efficiency metric, an agent that narrates a board in natural language and then guesses is competing against a human who can simply see the answer. The ceiling is where the points are, and you don't reach it by being verbose. It also makes the runs deterministic: same agent, same game, same 80 actions, every time.

The hardest part is seeing, not solving

Almost every failure was a confident, plausible reading of the board that was wrong about the board, and none of them raised an error. The agent would report something sensible, act on it, and walk somewhere it couldn't go.

The clearest example cost a full day on bp35. Every probe read the board at reset and treated it as fixed. On that reading the level is unwinnable: click every visible block and the exit never opens.

> **The world is bigger than the 64×64 frame**
> Visible blocks go from 147 to 273 on the fourth step right. Two blocks required by the solution are simply not on screen when the level starts.

The reason we didn't see it is worth more than the fix. It had been raised early that the whole level might not be visible, but that signal was argued down because the camera reported x=0, y=0 and never changed. That measurement was precise and irrelevant. The camera object didn't move; the content did.

Once probes re-scan after movement, bp35 level 1 is 15 actions against a human baseline of 21: the score cap. This was verified by replaying the sequence in our own build; the scorecard above predates it.

Other failures in the same family:

- A sprite sat on a tile whose background used the same colour value as "wall", so the agent reported being walled in on all four sides on an empty floor.
- Measurements taken every half-tile aliased: from one position it saw a block, five cells later a wall in the same place. Both readings were internally consistent.
- A move was declared impossible after being “measured every way it could be”, but every measurement had parked the relevant object exactly one cell out of reach.
- Buttons were recorded as inert after being tested from a single state. They were the movement controls; they simply did not respond until the machine entered a different configuration.

> **Core failure pattern**
> Exhaustive over what was sampled was being reported as exhaustive over what exists.

When the instrument is the bug

A separate class of failure is worse in a different way: the measurement apparatus corrupts the thing it measures while continuing to emit plausible numbers.

Row 63 of the bp35 board is an action counter. It gains a cell every action. Our reasoner hashed the raw grid to decide whether it had seen a state before, so every state was novel by construction. One run proudly reported 40 distinct states in 40 actions. One hundred percent new ground is not a good result; it is impossible.

We misidentified the player four times. Three times it was mistaken for that same action bar and once for static decoration. Conservation alone was not enough; the useful discriminator was translation: the player is the blob with the same shape in a new position.

ACTION7 is undo. It was in ARC’s datasheet. Our harness had classified it as a wasted action, so a free probe - try something, inspect the result, undo it - sat unused while we spent real actions answering the same questions.

A UTF-8 byte-order mark blanked the first key in a .env file. PowerShell’s Out-File -Encoding utf8 inserted an invisible U+FEFF before ANTHROPIC_API_KEY, so the parser saw a different key name. The file looked correct to a human reader.

Three runs that produced numbers while measuring nothing

Each of these was nearly read as a result:

1. **A void A/B arm.** Fifteen of sixteen actions fell back to the deterministic reasoner, so the arm measured the fallback, not the model.

2. The BOM again. Ten attempts across both arms, $0.00 spent, byte-identical results. A paid model that costs nothing did not run.

3. A branch mismatch. The run script did not exist on the checked-out branch, the command failed instantly, and the "results" were stale files from an earlier run.

> **Evaluation guardrail**
> Refuse to print an A/B comparison when fallback usage exceeds a threshold. Print VOID instead. Any agent evaluation with a fallback path can otherwise report a clean number for a component that never ran.

What "hard" means here

Before claiming any agent result, we measured the floor on bp35 level 1:

Policy	Episodes	Solved
Random: moves + clicks	200	0
Random clicks only	200	0 (never died)
Random moves only	200	0 (died 66.5%)
Tabular Q-learning	1,500	0

> **2,100 episodes. Zero solves.**
> Q-learning still learned something: its death rate fell from 59.0% to 28.5%. It learned to survive, not to solve. The death signal was learnable from raw grid hashes; the solve signal was not.

What we're honest about

tr87 was a replay and is not any more. For a long time we opened four of its levels by reproducing a recorded human play-through, which demonstrates nothing and cannot reach a level nobody recorded. The score above is derived: the agent reads the board's own legend, works out the rules, and dials the answer. Levels 5 and 6 are ones no recording could have given us.

Most of that gain was our own waste, not new reasoning. Finishing the game moved tr87 from 25.99% to a projected 67%. Removing redundant measurement moved it from 67% to the cap without changing a single deduction. The agent had been reading every dial's full cycle before turning it, on every slot of every level, when the first match was provably the only one.

We over-reported that run before the card corrected us. A level can score 115 and four of tr87's do, so our scorer computed a game total of 109.3%. The API caps the game at human and returned 100.0%. The scorer was wrong, is fixed, and the number here is the card's.

Twenty of the twenty-five public games are untouched. One game we have looked at still resists even a single legal action because its controls share nothing with the games we have solved. Averaged over the full public set rather than the five we have played, these numbers are small.

ARC-AGI-3's difficulty isn't precision. Where we can identify a game's mechanic, we tend to beat the human baseline outright. It is recognising the game at all: every game teaches its own rules through its own first level, and an agent that arrives carrying the last game's assumptions gets nothing.

> **The open problem**
> Can a system recognize what kind of world it has entered without importing assumptions from the last one?

All scores above were returned by ARC Prize's own scorecard API when each run was closed. Runs are reproducible and used no language model.