THE TURING BENCH

User Guide

A complete guide to the bench — both machines, every instrument, every control, the presets, and the Learn course.

Turing Bench · every number is measured, nothing is decorative

Turing Bench is a workbench for the oldest computer there is: a tape, a head, and a handful of rules. You build a Turing machine, watch it run, rewind it, race it, and finally feed it — as data — to a Universal Turing Machine that runs it for you. Everything happens in your browser; nothing is uploaded.

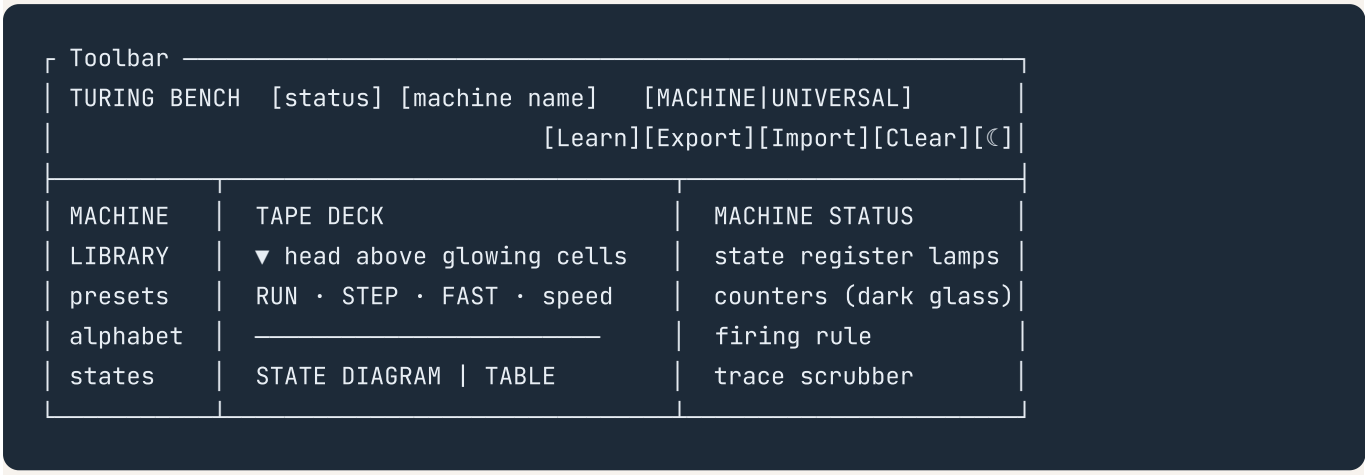
If Boole Bench showed you how machines are *built*, Turing Bench shows you what machines can *do* — and the one thing they provably cannot.

0. First visit: the welcome and the hints

On your very first visit the bench introduces itself — what a Turing machine is, what you can do here, and three ways to start: **Take the course** (lessons + hints), **Look around with hints**, or **Just let me play**. Reopen it anytime with the **?** button in the toolbar.

Hints are the plain-language layer: with the toolbar's **Hints** ◉ toggle on, hovering almost any instrument — a counter, a lamp rack, a button, a tape zone — shows a short explanation card. With hints off, the same texts remain available as ordinary hover tooltips. The first time you switch to UNIVERSAL mode, a one-time card explains the blue/amber color rule before you're left alone with it.

1. The bench at a glance



The bench opens in **daylight** (light theme). The ☾/☀ button toggles the dark bench; your choice is remembered. One thing never changes: tape cells, counters and lamps keep their dark instrument glass in both themes — real instruments do.

On screens narrower than 1100px the side rails become drawers — open them with ≡ **Library** and **Status** in the toolbar.

2. The three parts of every machine

- **Tape** — the strip of glass cells. Infinite in both directions: drag it sideways to pan, and blank cells (`·`) fade in forever. Click any cell to cycle its symbol through the alphabet. Edits made before the first step become the machine's saved input (RESET restores them); edits made mid-run are live experiments.
- **Head** — the metal assembly riding the tape. Per step it reads one cell, writes one cell, and moves one cell left or right (or stays).
- **State** — the single lit lamp in the STATE REGISTER. That lamp is the machine's entire memory.

3. Programming: the transition table

Open the **Transition Table** tab. Each row is one rule of the transition function δ:

> δ(state, read) → (write, move, next)

"In *state*, reading *read*: write *write*, move *move*, switch to *next*."

- Every cell of a row is a dropdown — click and change it in place.
- **+ Rule** adds a row for the first (state, read) pair you haven't covered.
- ✕ deletes a row. The row about to fire glows amber (● **firing next**).
- The footer counts uncovered pairs. An uncovered pair is not an error: if the machine reaches it, it simply halts with "no rule" — honestly stuck.
- Moving a row onto an occupied (state, read) pair replaces that pair's rule: the table always stays deterministic.

States and symbols live in the left rail: **+ State** and **+** (alphabet) add them, the ✕ on hover removes them — along with every rule that used them. The blank symbol `␣` cannot be removed.

4. The state diagram

The **State Diagram** tab shows the same δ as a machine you can walk through: states as instrument dials, rules as wires labelled `read / write, move`. The current state glows; the rule that fires next runs marching ants along its wire — exactly like a hot wire in Boole Bench.

The diagram is a *projection* of the table — edit either, both update:

- Drag from a node's right edge to another node → creates a rule (for the first symbol the source state doesn't handle yet; refine it in the table).
- Double-click empty canvas → new state. Select + Backspace deletes nodes or edges (and their rules).
- Drag nodes to tidy up — positions stick per machine. `⊡` re-fits the view.

5. Running: the control deck

Control	What it does
RUN (round button)	Runs at the fader's speed. Press again to pause.
STEP	Exactly one δ application, animated.
FAST	Headless run: engine only, no animation, live step counter. For busy-beaver-scale machines. Press STOP to interrupt.
Speed fader	1–30 steps per second.
RESET	Tape back to the saved input, state back to start, trace cleared.

Trace (right rail): every animated step is recorded. Drag the trace track to rewind the machine to any earlier step — tape, head and state all travel back — then scrub forward again, or STEP from the past (which discards the undone future, like typing after undo). FAST runs don't record a trace.

Runaway guard: after 10,000 steps without halting the bench pauses and warns. It does not claim the machine loops — it says *it cannot know*. Raise the guard ×10 and continue as often as you like. The lesson **The Limit** explains why no bench, however clever, could do better.

6. The machine library

Fifteen machines, each verified by the test suite. (They are *machines*, not programs — a Turing machine's rule table IS the machine. "Program" is reserved for UNIVERSAL mode, where a machine is encoded as data on the universal tape.) Click a machine to load it; the active one shows its full description in the rail, and every entry has a hover tooltip.

Machine	What it teaches
Bit Inverter	one state, one pass — the smallest honest machine
Binary Increment	carry propagation — the default machine
Binary Decrement	borrow rippling; 0 − 1 lights the UNDER lamp
Unary Addition	tape as memory: 111+11 → 11111
Two-Track Adder	two numbers stacked in one alphabet (□ ▀ ▄ ▉) — added in a single pass; tracks are convenience, not power
Palindrome Check	marking, back-and-forth head travel, YES/NO halts
Copy Machine	sub-routine thinking: 111 → 111_111
Bracket Matcher	pairing structure: balanced () → YES
0^n1^n Checker	the classic non-regular language — what tapes buy over finite automata
Busy Beaver 3	champion small machine: 6 ones in 14 steps
Busy Beaver 4	Brady's champion: 13 ones in 107 steps — STEP for chaos, FAST for the score
Busy Beaver 5	⚠ 47,176,870 steps — **use FAST, not RUN** (18 days at 30 steps/s). Ships with a 100M guard; proven optimal only in 2024
The Wrapper	Binary Increment with its input built in — the reduction gadget from the lesson The Domino Chain
Turing's Printer	never halts yet does real work — the productive opposite of a loop
Infinite Loop	never halts — trips the runaway guard on purpose

Loading a preset replaces the bench. Your own work survives via **Export** — a readable `.turingbench.json` file with the machine, its alphabet, states, rules and input. **Import** loads such a file (invalid files are rejected with a reason). **Clear** empties the bench to `q0` + `HALT` after asking.

7. UNIVERSAL mode

Flip the toolbar switch to **UNIVERSAL** and the bench rebuilds around a machine that runs machines. Whatever was on the bench becomes the **guest**, encoded onto the universal tape. Everything follows one color rule:

> **Blue is the universal machine. Amber is the guest it simulates.**

- **PROGRAM δ (encoded)** — the guest's entire rule table as blue `0`s and `1`s. The legend in the left rail is the whole dictionary: state q_i → i+1 zeros, symbol s_j → j+1 zeros, moves L/R/— → 0/00/000, `1` between fields, `11` between rules. A machine is just a string.
- **STATE REG** — the guest's current state, encoded, rewritten every cycle.
- **WORK TAPE** — the guest's amber tape. The **dashed** marker is the simulated head: dashed because it is not hardware, it is data. The solid blue head is the universal machine's own — watch it commute between zones.
- **δ-DECODER** — the encoded rules with live verdicts while the universal machine hunts: `state ≠` (rejected), `● matching`, `queued`.
- **μ-PROGRAM** — the universal machine's fixed five-phase program: FETCH → MATCH → WRITE → MOVE-SIM → RETURN. It is sealed but not secret: click the seal to inspect the actual μ-operation script of the current cycle. Nothing is fake.
- **STATE DIAGRAMS tab** — both machines' diagrams at once: the μ-machine's fixed five-state ring in solid blue (it never changes, whatever the guest), and the guest's diagram below in **dashed** rings — those states live on the tape as data, not in hardware. The current μ-phase and the guest's current state glow in their own colors, and the transition being executed runs its marching ants.

Two step buttons, two granularities:

- **μ-STEP** — one operation of the universal machine (read one cell, compare one cell, …).

- **SIM STEP** — run μ-steps until the guest advances exactly once.

The **COST OF UNIVERSALITY** readout keeps the score — "1 sim step ≈ N μ-steps". That N is measured, not staged: generality is bought with time, and every virtual machine and interpreter since 1936 pays the same bill.

8. The Learn layer

The **Learn** ◉ button opens the lesson panel — fifteen lessons in three acts, from "what is this thing" to the halting problem and the reductions it spawns (the current act is shown above every lesson title):

- **Act I · The Instrument** — 1. The Machine · 2. Reading the Program · 3. Your First Run · 4. Write Your Own
- **Act II · Judges and Runaways** — 5. Down and Under · 6. Machines That Judge · 7. More Than a Pattern · 8. Loops · 9. Two Kinds of Forever · 10. Busy Beavers
- **Act III · The Summit** — 11. Many Machines, One Power · 12. Machines as Data · 13. The Universal Machine · 14. The Limit · 15. The Domino Chain

The course ends with a hand-off: *the machine runs — what does the tape actually carry?* That question belongs to the next bench in the family, **Shannon Bench** (linked from the final lesson).

Lessons set the bench up for you (loading presets, switching modes) and ask you to **predict before you step** — what will the head write? Your answers are remembered, but nothing is ever locked: the prediction is offered, the step is never gated. Progress lives in your browser.

If you want the same course as pen-and-paper exercises, see the **Workbook**.

9. Tips & troubleshooting

- **The machine halted and RUN is dead** — that's what HALT means. RESET.
- **"Halted · no rule"** — the table footer names the missing (state, read) pair. Add a rule for it, then RESET.
- **The FAST counter froze at a round number** — that's the runaway guard. Raise it ×10 in the warning box, press FAST again.
- **I scrubbed back and want my future back** — scrub forward; it's all there until you STEP from the past.
- **My import is rejected** — the message names the first problem (unknown state, duplicate rule, symbol outside the alphabet…). Files are plain JSON; open them in any editor.
- **Everything is tiny on my phone** — running works on touch; *building* machines is desktop-first, by design.