Why Φ strictly grows from one era to the next

A walk-through of the proof for the sweeper Turing machine

`1RB1LA_1RC0RF_1RD---_0LE1RB_---0LA_1LD1RF`

What we want to prove

The Turing machine under study runs forever in an obvious empirical pattern: the tape is repeatedly "swept" to the right; whenever a sweep finishes, the machine "rebounds" back to the left, dumps a few new 1s on the left tape, and starts a new sweep. Each cycle of *sweep right + rebound back to a fresh sweep* is called an **era**. We label the configuration at which each era begins $e_0, e_1, e_2, \ldots$.

Each configuration has a tape that, in our compact "run-length" notation, looks like

$$\big[\, a_1, a_2, \ldots, a_k \,;\, c \,;\, b_1, b_2, \ldots, b_m \,\big] \qquad \big(a_i \geq 1,\ b_j \geq 1,\ c \geq 0\big),$$

where the a's record the lengths of the 1-runs to the left of the cursor, c is the length of the run *at* the cursor, and the b's are the runs to the right.

We define a single number that measures the "mass" of such a tape:

$$\boxed{\Phi \;:=\; (a_1 + a_2 + \cdots + a_k) \;+\; (b_1 + b_2 + \cdots + b_m) \;+\; c\,.}$$

Empirically, simulating the machine for billions of steps shows

$$\Phi(e_0) < \Phi(e_1) < \Phi(e_2) < \cdots$$

and in fact $\Phi(e_{n+1}) \geq \Phi(e_n) + 4$. The point of this document is to explain, line by line, why this is provable from the macro rules alone, with no reference to simulation.

The proof has been formalised in Lean 4. The names of the Lean lemmas appear in `this font` so the reader can locate each step in the source. The relevant files are:

`phi.lean`	Per-rule arithmetic (one lemma per macro rule).
`phi_progress.lean`	Lift to one "macro step" and to a whole era.
`phi_era.lean`	The headline theorem: Φ jumps by ≥ 4 across an era.

1 The macro configuration

In the formalisation a configuration is one of two shapes:

$$M(L, c, R) \qquad \text{or} \qquad M_0(L, R),$$

where L, R are lists of natural numbers (the a_i's and b_j's), and $c \in \mathbb{N}$ is the cursor count. The M_0 form is a transient "zero-cursor" state used between rules; it has no explicit cursor count, and we treat its cursor as 0 for the purpose of Φ. Hence, in Lean,

$$\Phi\big(M(L, c, R)\big) = \mathrm{sum}(L) + \mathrm{sum}(R) + c, \qquad \Phi\big(M_0(L, R)\big) = \mathrm{sum}(L) + \mathrm{sum}(R).$$

This is `MacroConfig.phi` in `phi.lean`, lines 32–40.

2 The macro rules and what they do to Φ

The machine has been analysed into 18 elementary "macro rules". Each rule matches a small pattern in a configuration and rewrites it deterministically. We list them grouped by family, alongside the *change in* Φ that the rule causes; this change is denoted $\Delta\Phi$.

To prove $\Delta\Phi$ is what we claim, in each case we just substitute the matched and rewritten configurations into the definition of Φ and simplify with elementary arithmetic. Each such substitution is one short Lean lemma in `phi.lean`.

2.1 Sweep family (4 rules) — $\Delta\Phi = 0$

These rules move the cursor one step right while simultaneously decrementing the cursor count by 2 and bumping a neighbouring run. Mass is conserved.

- **Sweep:** $M\big(a{::}L,\ c{+}3,\ d{::}R\big) \to M\big((a{+}1){::}L,\ c{+}1,\ (d{+}1){::}R\big)$.

  Compute:

$$\begin{aligned}\Phi(\text{out}) &= (a{+}1) + \text{sum}(L) + (d{+}1) + \text{sum}(R) + (c{+}1)\\ \Phi(\text{in}) &= a + \text{sum}(L) + d + \text{sum}(R) + (c{+}3)\\ \Delta\Phi &= (a{+}1{+}d{+}1{+}c{+}1) - (a{+}d{+}c{+}3) = 0.\end{aligned}$$

  Lemma: `phi_macro_sweep`.

- **SweepL, SweepR, SweepS**: the same calculation when one or both of L, R are empty.

  Lemmas: `phi_macro_sweep_left_empty`, `..._right_empty`, `..._solo`.

2.2 SweepE family (4 rules) — $\Delta\Phi = 0$

When the cursor count reaches the special value 2, the same kind of sweep is performed but the configuration drops into the M_0 form (cursor count vanishes; or formally, becomes 0):

$$M\big(a{::}L,\ 2,\ d{::}R\big) \ \to\ M_0\big((a{+}1){::}L,\ (d{+}1){::}R\big).$$

Computing Φ:

$$\Phi(\text{out}) = (a{+}1) + \text{sum}(L) + (d{+}1) + \text{sum}(R) = a + \text{sum}(L) + d + \text{sum}(R) + 2 = \Phi(\text{in}).$$

Lemmas: `phi_macro_sweep_to_zero`, `..._left_empty`, `..._right_empty`, `..._solo_to_zero`.

2.3 Shift (1 rule) — $\Delta\Phi = 0$

When the cursor count is exactly 1 (transient case), the cursor "hops" across the run boundary:

$$M\big((a{+}1){::}L,\ 1,\ d{::}R\big) \ \to\ M\big(L,\ a{+}1,\ 1{::}d{::}R\big).$$

Both sides give $\Phi = a + 1 + \text{sum}(L) + d + \text{sum}(R) + 1$. Lemma: `phi_macro_shift`.

2.4 EraDone (1 rule) — $\Delta\Phi = +4$

This is the rule that is responsible for the strict increase. When the right side has shrunk to a single 1, the machine rebounds back left and dumps the leftmost run plus a fresh cursor of size $a + 6$:

$$M_0\big((a{+}1){::}L,\ [1]\big)\ \to\ M\big(L,\ a{+}6,\ []\big).$$

Computing:

$$\begin{aligned}\Phi(\text{out}) &= 0 + \text{sum}(L) + (a{+}6),\\ \Phi(\text{in}) &= (a{+}1) + \text{sum}(L) + 1,\\ \Delta\Phi &= (a{+}6) - (a{+}2) = +4.\end{aligned}$$

Lemma: `phi_macro_era_complete`.

2.5 Bounce/Two/Multi families (8 rules) — $\Delta\Phi = +2$

Whenever the right side begins with a run of length ≥ 2, the cursor "bounces" and the rule injects new 1s. In every case, the head 1 and a constant from the rule combine to give exactly $\Delta\Phi = +2$.

For instance, **Bounce**:

$$M_0\big(a{::}L,\ [z{+}4]\big)\ \to\ M\big((a{+}4){::}L,\ z{+}1,\ [1]\big).$$

Compute:

$$\begin{aligned}\Phi(\text{out}) &= (a{+}4) + \text{sum}(L) + 1 + (z{+}1),\\ \Phi(\text{in}) &= a + \text{sum}(L) + (z{+}4),\\ \Delta\Phi &= (a{+}4{+}1{+}z{+}1) - (a{+}z{+}4) = +2.\end{aligned}$$

The other rules in this group (**BounceE**, **Two**, **TwoS**, **Multi2**, **Multi2E**, **MultiN**, **MultiNE**) are analogous — each one dumps a constant amount of fresh mass that totals to +2. The eight lemmas in `phi.lean` (`phi_macro_zero_bounce`, ..., `phi_macro_multi_bounce_general_to_zero`) verify the count.

2.6 Summary table

Family	Rules	$\Delta\Phi$
Sweep	4 rules	0
SweepE	4 rules	0
Shift	1 rule	0
EraDone	1 rule	+4
Bounce/Two/Multi	8 rules	+2

Key observation. Every macro rule satisfies $\Delta\Phi \geq 0$. Moreover, the only rule that fires when the configuration has the shape $M_0(L, [1])$ — i.e. a zero-cursor with a single 1 on the right — is `EraDone`, whose $\Delta\Phi$ is *exactly* +4. (Cf. `macroStep_phi_strict_at_era`.)

3 One macro step

The 18 rules above are wrapped into a single dispatch function `macroStep` that, given a configuration cfg, picks the unique rule whose left-hand side it matches and returns the rewritten configuration cfg′ together with a step count k. (Some "stuck" configurations have no matching rule; for these `macroStep` returns nothing.)

Theorem 1 (`macroStep_phi_nondec` in `phi_progress.lean`)**.** *For any configurations* cfg *and* cfg′ *such that* ***macroStep***$(\text{cfg}) = \text{some}(k, \text{cfg}')$*,*

$$\Phi(\text{cfg}) \le \Phi(\text{cfg}').$$

Proof. The function `macroStep` is a giant case analysis on the shape of cfg. There are roughly two dozen branches; in each branch either no rule fires (and the hypothesis is impossible), or exactly one rule fires, in which case the per-rule lemma from Section 2 directly gives $\Delta\Phi \in \{0, +2, +4\}$, all of which are ≥ 0.

In Lean this is written as a structural case analysis on cfg $= M(L, c, R)$ vs. $M_0(L, R)$, then on whether L and R are empty or non-empty, then on the small values of the cursor c. Each of the dispatched branches calls a single tactic block of the form

```
ms_simp at h; obtain ⟨rfl, rfl⟩ := h;
simp only [phi_M, phi_M0, sum_cons, sum_nil]; omega.
```

The first line forces Lean to read off which rule fired (by unfolding `macroStep`). The second unfolds Φ into a pure arithmetic expression. The third (`omega`) closes the resulting linear inequality automatically. □

Theorem 2 (`macroStep_phi_strict_at_era` in `phi_progress.lean`)**.** *If the input has the era-end shape* $M_0(L, [1])$ *and* ***macroStep*** *fires, then*

$$\Phi(\text{cfg}') \;\ge\; \Phi(\text{cfg}) + 4.$$

Proof. We just specialise the case analysis. When the input is $M_0(L, [1])$, the only firing branches of `macroStep` are `EraDone` (if L has at least one element, the leading run of which is at least 1; formally $L = (a{+}1) :: L'$) or its solo variant. Both have the explicitly computed $\Delta\Phi = +4$ from Section 2.4. Lean closes the case as before, with `ms_simp; obtain ...; simp; omega`. □

4 From one step to a whole era

We now bundle steps into eras. The function `macroEra` in `progress.lean` is simply "apply `macroStep` up to fuel times, summing the step counts":

$$\texttt{macroEra}(0, \text{cfg}) = (0, \text{cfg}) \quad \text{and} \quad \texttt{macroEra}(\text{fuel}{+}1, \text{cfg}) = \begin{cases} (0, \text{cfg}) & \text{if } \texttt{macroStep}(\text{cfg}) = \text{none}, \\ (k + k', \text{cfg}'') & \text{otherwise}, \end{cases}$$

where in the second case $\texttt{macroStep}(\text{cfg}) = (k, \text{cfg}')$ and $\texttt{macroEra}(\text{fuel}, \text{cfg}') = (k', \text{cfg}'')$.

Theorem 3 (`macroEra_phi_nondec` in `phi_progress.lean`)**.** *For any natural number* fuel *and any configuration* cfg*,*

$$\Phi(\text{cfg}) \;\le\; \Phi\big(\textit{macroEra}(\text{fuel}, \text{cfg}).2\big),$$

where ***macroEra***$(\ldots).2$ *denotes the resulting configuration.*

Proof. Induction on fuel.

*Base (*fuel $= 0$*).* The result is just cfg itself, so the inequality $\Phi(\text{cfg}) \le \Phi(\text{cfg})$ is trivial.

Step. Suppose the result holds for fuel$'$ and consider fuel $=$ fuel$' + 1$. Two sub-cases:

- If `macroStep`(cfg) = none, then by definition `macroEra` returns cfg unchanged; the inequality is again trivial.
- Otherwise, `macroStep`(cfg) $= (k, \text{cfg}')$. Then by Theorem 1, $\Phi(\text{cfg}) \le \Phi(\text{cfg}')$. By the induction hypothesis applied to cfg$'$, $\Phi(\text{cfg}') \le \Phi\big(\texttt{macroEra}(\text{fuel}', \text{cfg}').2\big)$. Chaining these via transitivity of $\le$ gives the conclusion.

□

5 Era-start configurations

An *era-start* configuration is a configuration of shape $M(L, c, [1])$ with $L \neq []$, all runs ≥ 1, and $c \ge 4$. This shape captures the empirical era-boundary form. In Lean it is the record `EraStartConfig` (file `era.lean`), with embedding $e \mapsto e.\texttt{toMacro} = M(e.L, e.c, [1])$.

The crucial *sequential* fact about era-starts is the following: once you start an era from e, the trajectory under `macroEra` proceeds through (a) some number of sweep / Shift steps that each have $\Delta\Phi = 0$, then (b) one final $M_0(L', [1])$ shape, from which (c) the next `macroStep` fires `EraDone` (or its solo variant) and produces the *next* era-start e'.

This intra-era story is developed in `era.lean` (theorems `macroStep_M_intra_era_preserves_L_ne_nil`, `macroStep_M0_R1_produces_era_start`, etc.). For the purposes of this document, all we need is that there exists some fuel and some list L such that

$$\texttt{macroEra}(\text{fuel}, e.\texttt{toMacro}).2 \;=\; M_0(L, [1]),$$

and that one further `macroStep` from $M_0(L, [1])$ produces $e'.\texttt{toMacro}$.

6 The headline theorem

We can now combine everything.

Theorem 4 (`phi_strict_across_era_step` in `phi_era.lean`)**.** *Let e be an era-start configuration. Suppose*

$$\textit{macroEra}(\text{fuel}, e.\textit{toMacro}).2 = M_0(L, [1])$$
$$\textit{macroStep}\big(M_0(L, [1])\big) = (k,\ \text{cfg}').$$

Then

$$\Phi(\text{cfg}') \;\ge\; \Phi(e.\textit{toMacro}) + 4.$$

Proof. We have a chain of inequalities. By Theorem 3 applied to cfg $= e.\texttt{toMacro}$,

$$\Phi(e.\texttt{toMacro}) \;\le\; \Phi\big(M_0(L, [1])\big). \tag{$\star$}$$

By Theorem 2,

$$\Phi\big(M_0(L, [1])\big) + 4 \;\le\; \Phi(\text{cfg}'). \tag{$\star\star$}$$

Adding 4 to both sides of ($\star$) and chaining with ($\star\star$):

$$\Phi(e.\texttt{toMacro}) + 4 \;\le\; \Phi\big(M_0(L,[1])\big) + 4 \;\le\; \Phi(\mathrm{cfg}'),$$

which is the conclusion. (In Lean: a single `omega` call closes the chain, given the two named hypotheses.) □

Corollary 5 (`phi_strict_between_era_starts`)**.** *If, in addition,* $\mathrm{cfg}' = e'.\texttt{toMacro}$ *for some era-start configuration* e'*, then*

$$\Phi(e'.\texttt{toMacro}) \;\ge\; \Phi(e.\texttt{toMacro}) + 4.$$

Proof. This is the same statement as Theorem 4 specialised to the case where the post-step configuration happens to be an era-start. (One can show by separate analysis that the post-step configuration is always an era-start; that work lives in `macroStep_M0_R1_produces_era_start` in `era.lean`.) □

This is the statement the user observed empirically:

$$\Phi(e_{n+1}) \;\ge\; \Phi(e_n) + 4.$$

In particular, Φ is strictly increasing along the era boundaries, which is the rule the user noticed in simulation.

Recap of the logic

The proof is layered:

1. Each macro rule, in isolation, has a $\Delta\Phi$ that we compute by direct substitution. There are 18 of these, one per rule (Section 2). Every $\Delta\Phi$ is 0, 2, or 4.
2. One `macroStep`, by case analysis on which rule fires, gives $\Delta\Phi \ge 0$ (Theorem 1). When the input is the era-end shape $M_0(L,[1])$, the only firing rule is `EraDone`, whose $\Delta\Phi = +4$ exactly (Theorem 2).
3. Many `macroStep`s in a row preserve $\Phi \ge$ start (Theorem 3, by induction).
4. Composing (3) with (2) gives the strict $+4$ jump across one era (Theorem 4).

The Lean files `phi.lean`, `phi_progress.lean`, and `phi_era.lean` contain no `sorry`: the proof is mechanically checked from start to finish.