

Not peer-reviewed and not journal-published. Draft v5 (2026). Content may change. © José Luis D'Andrade · IdentIA / Self-Improving Society. **All rights reserved.** Brief citation with attribution is permitted. **Reproduction, translation, adaptation, or reuse**, in whole or in part (including commercial use), is **prohibited** without the **copyright holder's prior written permission**.

The formulas of the systems paper, explained

Self-Improving Society — Paper v5.0 (systems). Each formula as it appears in the paper, followed by an explanation for non-technical readers and, where the name deserves it, where it comes from and why it is called that.

§3.1 · The identity object and the measurement

$$\Psi(p) = \langle \text{SOUL}(p), M(p), L(p), K(p) \rangle$$

$$m_t(p) = \langle R_t, \theta_t \rangle$$

These two formulas answer a question an auditor asks in any institution: **if we change the AI model an agent uses, is it still the same agent?**

The first says what an agent **is**. Its identity, written Ψ , is made of four things that are kept over time: its founding charter (SOUL: what it exists for and how it must behave), its memory (M: what it has learned), its lineage (L: where it comes from and which versions it has had) and its skills (K: what it knows how to do, and where each skill came from). It is what makes an employee the same person after a change of computer: their record, their training and their post do not depend on the machine.

The second says what a **measurement** of the agent is at a given moment: which engine it is running on now (R_t) and what level of skill we are measuring today (θ_t). That is the computer and this quarter's performance review: they change, and the person does not change because of it.

The practical consequence is the one that matters to whoever signs: **changing model vendor does not change the agent's identity**, and therefore does not break the traceability of what it did before. And the reverse: if someone alters the charter, the memory or the lineage, it is no longer the same agent, and the system treats it as such.

Where it comes from. Nowhere to cite: these two formulas are the paper's own contribution (Definitions 1 and 2 of §3.1). The nearest prior work measures identity properties of language-model agents or lists identity as a requirement; none gives identity as a persistent object separated from its realization, which is what these two formulas do. **Why Ψ (psi).** It is the Greek letter physics uses for the state of a system. The paper borrows it with that single meaning, "the state that persists", and says expressly that it relies on no other reading from physics: it is notation, not analogy.

§5.1 · Variation and selection

$$\text{RSI} = \underbrace{\text{Variation}}_{\text{exists (runtime)}} \circ \underbrace{\text{Selection}}_{\text{this ladder}}$$

This formula says what **self-improvement** (RSI, for *recursive self-improvement*) is made of: two distinct operations, chained.

The first is **variation**: producing changes. Agents already do this on their own: they write new skills, fix the ones they have, summarize what they learned. It is like a staff that proposes new ways of doing the work every week. That already exists and did not have to be built; the agent engine brings it.

The second is **selection**: deciding which of those changes are actually better and deserve to stay, and which are discarded. That is what almost every system lacks, and it is what this project builds: the "ladder" the following sections describe.

The symbol in the middle, the circle, means "one after the other": first vary, then select, and the result enters the cycle again. The sentence that accompanies the formula in the paper is the key: **variation without selection is drift** (the system changes, but nobody knows whether for the better), and **selection without variation has nothing to choose from**.

For a decision-maker the reading is this: a system that "learns by itself" without an explicit selection mechanism is not improving, it is changing. The difference between the two is who judges the change, and with what evidence. Everything after this formula is the answer to that question.

Where it comes from. From Darwin. The paper says so plainly: improvement is never declared by the thing that changed; it is selected by measured comparison against alternatives, in an environment. Variation and selection are the two engines of *On the Origin of Species* (1859), and the formula does no more than write them side by side.

§5.4 · F0 — Beta–Bernoulli reputation

$$\pi \sim \text{Beta}(\alpha, \beta), \quad \alpha = s + 1, \quad \beta = (n - s) + 1, \quad \mathbb{E}[\pi] = \frac{\alpha}{\alpha + \beta}$$
$$\hat{p}_{\text{LB}} = \frac{\hat{p} + \frac{z^2}{2n} - z \sqrt{\frac{\hat{p}(1 - \hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}, \quad \hat{p} = \frac{s}{n}$$

These two formulas solve a problem any department head recognizes: **how do you compare someone who has been here a week with someone who has been here a year?**

A new agent has done one task and it went well: 1 out of 1, 100%. Another has done a hundred and got ninety right: 90%. If you rank by percentage, the newcomer comes first. Nobody with judgment would do that, because a single success says almost nothing. The problem is that a computer, unless told otherwise, does exactly that.

The first formula fixes the error at the root: instead of storing a percentage, it stores **a distribution**, that is, the percentage and how much confidence there is in it. With s successes in n attempts, the Beta formula produces a curve: narrow and sharp for the veteran (we know well what they are worth), wide and flat for the newcomer (they could be worth almost anything). The expectation, $E[\pi]$, is the central value of that curve.

The second formula, Wilson's, is the decision rule: **rank not by the central value but by the worst plausible value**. For the veteran that worst value is high, because there is a lot of evidence; for the newcomer it is low, because there is little. So 90/100 ranks above 1/1, which is what common sense would dictate.

What this buys the institution is automatic prudence: **nobody rises in rank on a stroke of luck**, and a newly created agent has to prove its worth before the system entrusts it with important work. It is the first rung of the ladder and the only one that already works with no more data than a count of successes.

Why it is called "Beta–Bernoulli reputation", and where it comes from. The name joins two surnames from classical statistics. **Bernoulli** (Jacob Bernoulli, 17th century) names the simplest experiment there is: something that goes right or wrong, with no middle ground, like tossing a coin or solving a task. Every task an agent does is a "Bernoulli trial": success or failure. **Beta** is the distribution statisticians have used for two centuries to answer the natural question that follows: *after seeing s successes in n attempts, what is this coin's probability of success, and how sure am I?* The two fit exactly: if you start knowing nothing (a flat Beta) and keep adding successes and failures, the curve sharpens around the true rate, and the formulas $\alpha = s + 1$ and $\beta = (n - s) + 1$ are literally "successes plus one, failures plus one". It is the same mechanism online marketplaces use to rate sellers, and for the same reason: a seller with a single five-star review cannot rank above one with a thousand reviews averaging 4.8. Calling it **reputation** in the paper is deliberate: what is being computed is not a grade but how much trust an agent deserves in view of its record, with the uncertainty built into the number itself. The **Wilson bound** is named after Edwin Wilson, who published it in 1927; it is the standard correction for not trusting percentages computed over few cases.

§5.5 · F1 — Bradley–Terry and TrueSkill: compare, do not score

$$P(i \succ j) = \frac{\gamma_i}{\gamma_i + \gamma_j} = \sigma(\beta_i - \beta_j)$$

$$\gamma_i \leftarrow \frac{\sum_{j \neq i} w_{ij}}{\sum_{j \neq i} \frac{n_{ij}}{\gamma_i + \gamma_j}}$$

$$\mu_i \leftarrow \mu_i + \frac{\sigma_i^2}{c} v\left(\frac{\mu_i - \mu_j}{c}\right), \quad \sigma_i^2 \leftarrow \sigma_i^2 \left[1 - \frac{\sigma_i^2}{c^2} w\left(\frac{\mu_i - \mu_j}{c}\right)\right]$$

The previous rung measures each agent on its own: how many tasks it got right. This rung does something different and more demanding: **it compares them with each other**. The reason is that "got it right" depends on the task: an agent that succeeds at easy tasks is not better than one that succeeds at hard ones. Comparing two agents on the same task removes that bias, just as in a tournament what matters is not your total points but whom you beat.

The first formula (Bradley–Terry) says how a list of head-to-heads becomes a ranking. Each agent has a "strength" y , and the probability that i beats j is the strength of i divided by the sum of the two. It is exactly the model used by the public Chatbot Arena leaderboard of language models, and the same principle as chess Elo: beating a strong player is worth more than beating a weak one.

The second formula is the procedure for computing those strengths from the results. It is called MM iteration and has a practical virtue the paper stresses: it can be written in a few lines with no external library. That matters to whoever audits: the ranking can be recomputed by hand and checked; it does not come out of a black box.

The third (TrueSkill) adds what Bradley–Terry lacks: **uncertainty**. Each agent has a mean μ (how good it is) and a deviation σ (how sure we are of that), and every result moves both: the mean up or down, the uncertainty always down, because every match teaches something.

What the institution gains is a ranking that resists the most common trick: **picking easy tasks to look good**. Here only whom you beat counts.

Why they are called that, and where they come from. **Bradley–Terry** carries the surnames of two statisticians, Ralph Bradley and Milton Terry, who published it in 1952 in the journal *Biometrika* for a very concrete problem: how to rank products when all you have are pairwise comparisons, of the kind "the taster preferred A over B". Their idea was to give each thing a strength number and to say that the probability that A beats B is A's strength divided by the sum of the two. Seventy years later it is still the standard model for turning contests into a ranking: it is what lies behind chess Elo, the Chatbot Arena leaderboard and much of the machinery that learns from human preferences. The second formula, the "MM" iteration, is the method David Hunter proposed in 2004 for computing those strengths simply and stably; the project chose it because it fits in a few lines of code and can be recomputed by hand.

TrueSkill is not a surname but a product name: Ralf Herbrich, Tom Minka and Thore Graepel created it in 2006 at Microsoft Research for player matchmaking on Xbox Live. The problem it solved was practical: Elo works well with many games, but a new player needs dozens before their score is reliable, and is matched badly in the meantime. TrueSkill added to the score a **measure of uncertainty** that starts large and shrinks with every game, and extended the model to draws and teams. The name alludes to that: the visible score is an estimate of the real skill, and the system knows how far it is from it. The paper uses it for the same reason Xbox did: **do not trust a newcomer until the uncertainty has come down**.

§5.6 · F2 — Item Response Theory: the reliability gate

$$P(\text{profile } p \text{ solves task } k) = \sigma(a_k(\theta_p - b_k))$$

The two previous rungs share a crack: they look at the agent, not at the task. This rung closes it. The formula says that the probability that an agent solves a task depends on **two things at once**: the agent's ability (θ) and the task's difficulty (b), plus a third number (a) that measures how well that task tells good from bad. If the task is so easy anyone gets it right, it says nothing about the agent; if it is so hard nobody solves it, neither.

The consequence is what gives the section its name: it is the **reliability gate**. With this model an agent **cannot rise in rank by choosing easy tasks**, because each task's difficulty is estimated at the same time as each agent's ability, and getting easy things right is worth little. And the system can also tell whether its own estimates are stable: if an agent's ability changes a lot from one estimate to the next, the signal is not reliable, and **the system refuses to act on it**. Earlier versions of this note added that this is exactly what happened in the only production run the paper reports — the gate did not converge and everything above it stayed blocked. **That reading is retracted** (systems paper §7.3): the convergence test measured the per-iteration change before the step that centres item difficulties, so the flag could not report success for any data. The rule is still in the code; no live run has been observed applying it.

Why it is called that, and where it comes from. Item Response Theory was born in standardized testing. Its original problem was ours: two students who answer different questions cannot be compared by their number of correct answers, because the questions are not worth the same. The solution was to model each question ("item") with its own difficulty and to estimate each person's ability at the same time as each question's difficulty. Frederic Lord formulated it in the United States from the 1950s on, in parallel with the Dane Georg Rasch (1960); Lord's 1980 book, *Applications of Item Response Theory to Practical Testing Problems*, is the one the paper cites. The specific model in the formula is the "two-parameter" one (2PL): difficulty and discrimination per item. It is the mathematics behind tests such as the SAT, the GRE or PISA, and the paper uses it for the same purpose: **measure agents the way examinees are measured, discounting the question**.

§5.7 · The dependency that closes the ladder

$$\text{enable}(F5) \iff \text{Done}(F2) \wedge \text{Done}(F3) \wedge \text{Done}(F4)$$

This is not a formula for computing something but a **rule**, written in logic so that it admits no interpretation. It says: the system may enable recursive self-improvement ($F5$, the last rung, where the system edits itself) if, and only if, three conditions hold at once: that the reliability gate has converged ($F2$), that acting on the measurement is stable and reversible ($F3$), and that the diversity of agents is protected ($F4$).

The paper insists on a detail that changes everything: "done" does not mean "the code exists" but **"the condition has been measured live and holds"**. Earlier versions said that is why $F5$ is blocked today — not by a policy decision but because the $F2$ measurement returned *converged* = *False*. **That is retracted** (systems paper §7.3): the flag could not become `True` under the criterion then in use, and after the repair a fit in a development environment reports *converged* = *True*. $F5$ and $F3$ are off today **by decision**. What anyone can still check by reading the data is the rule itself, not a live application of it.

Where it comes from. The paper relates it to Gödel's second incompleteness theorem (1931): a sufficiently expressive system cannot prove its own consistency from within. It does not claim the theorem applies literally; it adopts its engineering moral: the guarantees that authorize a system to modify itself must sit **outside** the loop

they certify. A system that grades its own improvement with the same signal it is optimizing has no ground to stand on.

Appendix D.1 · The human analog and the Ω term

$$\hat{\Psi}_{\mathcal{M}}(p^H) = \langle S_{\mathcal{M}}(p^H), M_{\mathcal{M}}(p^H), L_{\mathcal{M}}(p^H), K_{\mathcal{M}}(p^H) \rangle, \quad p^H \not\equiv \hat{\Psi}_{\mathcal{M}}(p^H)$$

This formula is a warning written in symbols. It takes the same identity tuple of §3.1 and applies it to a **person** (p with superscript H, for *human*), seen through some model M: what that model captures of their character, their memory, their history and their skills. And then it states, with the "not equivalent" sign, what everyone knows but systems forget: **what a model captures of a person is never the person**. There is always a residue the model does not see, and that residue is called Ω .

The reason for writing it is design, not philosophy. If Ω were part of the identity Ψ , nobody could verify whether an identity had been preserved, because one of its pieces would be, by definition, unobservable. That is why the paper keeps Ω **outside** the identity object: identity is made only of what can be stored, inspected and enforced. What cannot be, is not promised.

Why Ω (omega). It is the last letter of the Greek alphabet, and many disciplines use it for "the remainder", "the limit" or "what lies outside". The paper chooses it in that sense: it is the formalism's own incompleteness term, the part the model admits it does not model.

Appendix D.2 · The behavior map

$$b_t(p) = f(\Psi(p), m_t(p)) + \varepsilon_t, \quad \varepsilon_t \sim \Omega(p)$$

The last formula ties the others together. It says what an agent's **observed behavior** at a given moment (b_t) depends on: its identity (Ψ , what persists), the measurement of the moment (m_t : the engine running it and the ability estimated for it), and a noise term (ε) that is not modelled and that comes from Ω .

Put differently: two runs of the same agent, with the same identity and the same engine, may behave differently, and the formula acknowledges it instead of hiding it. What matters is **where** that noise lives: in the behavior, not in the identity. Identity remains verifiable; what gets bounded is the effect of the noise, and that is governance's job, not the ontology's.

For a decision-maker this answers a common objection: "if the agent does not always behave the same, how can you say it is the same one?" The paper's answer is that **being the same and behaving the same are different things**, just as with people, and that the system only promises the first, which is the one that can be audited.