Runescript as Program: A Compositional and Traceability Framework for Contemporary Rune-Stave Construction

Anna Berelet — Independent Researcher — ORCID 0009-0009-7474-8410

Preprint — 2026

Abstract

Background. The modern runic revival has produced a rich repertoire of *runescripts* and bind-runes — composite rune-figures made for operative ends — yet the field lacks an explicit account of how such figures are *constructed* rather than merely *interpreted*. A recurrent mode in modern practical literature selects runes thematically ("by meaning") and arranges them for visual balance — an aesthetic emphasis rather than a generative one.

Contribution. We present a compositional *design* framework in which a runescript is modeled as a *program* rather than a static charm. A structured statement of intention — the Wish–Outcome–Obstacle–Plan (WOOP) model from goal-pursuit psychology — is compiled into a directed graph of runes treated as *typed operators* (force, control, modifier, guard, connector, and target/state roles), composed by a small grammar: sequence, binding, conditional handling, bounded iteration, guarding, and reusable sub-routines. Construction is checked at **two levels**: *structural well-formedness* (reachability, defined triggers, bounded loops, no unused elements, an observable target and recorded completion) and *interpretive adequacy* (each transition carries an articulated, non-thematic rationale).

Epistemic framing. The framework is set within an explicit layered epistemology separating historically attested practice, twentieth-century revival construction, documented psychological mechanism, and unverified metaphysical claims, under an "as-if" stance toward efficacy.

Evaluation and limits. The framework is evaluated **conceptually**, through worked examples and explicit well-formedness checks (including negative examples); **no empirical efficacy claim is made**, and the psychological literature is cited for the method's *components*, not for the rune-stave form. The contribution is the method — a reusable, inspectable construction framework, conceptually reproducible — while the specific implemented rune-to-operator mapping is withheld and not reproducible from this paper alone.

Keywords: runescripts; bind-runes; contemporary esotericism; runic revival; compositionality; visual symbolic language; implementation intentions; WOOP; traceability; conceptual methods paper.

---

1. Introduction

In the contemporary runic revival, practitioners routinely compose **runescripts** and **bind-runes** — composite figures assembled from individual runes and made for *operative* ends: to advance a goal, protect an undertaking, or hold an intention. A large literature instructs the practitioner in what each rune *means* and how a rune may be *interpreted* in divination (Thorsson 1984; Paxson 2005). Strikingly little of it addresses the converse and more basic question this paper takes up: by what logic is an operative figure *constructed*?

In a recurrent mode of practice the answer is implicit and aesthetic. Runes are chosen *thematically* — by the meaning each is felt to carry in relation to the wish — and then arranged for visual balance, so that the finished figure is harmonious. The governing emphasis is visual composition rather than any specified transition structure; the figure is read afterward for the meanings it gathers, but its composition does not specify how those meanings act in concert. In this paper we use **static talisman** as an analytical term for such a figure: a composite whose elements are co-present but for which no explicit transition structure is specified. (This is a descriptive label, not a claim that such figures lack their own ritual logic.)

This paper proposes an alternative: to model an operative runescript as a **program** rather than a static talisman. On this model a structured statement of intention is *compiled* into a directed graph of runes, where each rune occupies a position because it discharges a specific operational role — and where the finished figure can be *traced*, step by step, to confirm that it moves from an initial condition to a stated goal. Form follows the structure of the program (input → operations → output, with branches and loops). The framework does not impose symmetry as a compositional constraint; asymmetry may therefore emerge naturally where distinct branches encode distinct functions, but it is neither required nor a mark of authenticity.

The contribution is threefold: (i) a **typology** of runes as typed operators, with an inversion principle that doubles the operator set; (ii) a small **compositional grammar** — sequence, binding, conditional handling, iteration, guarding, and reusable sub-routines — over those operators; and (iii) a **two-level validation criterion** (structural well-formedness plus interpretive adequacy) that lets a candidate figure pass or fail explicit construction checks independently of any belief about magical efficacy. Together these turn an implicit, aesthetic craft into an explicit, inspectable construction method whose structural criteria are refutable.

Two boundaries frame the contribution. First, *epistemically* (§3): this is a description and formalization of a **modern constructive practice**, not a claim about historical magical technique and not a claim that the resulting figures alter the external world; every claim is tagged by layer, and the question of literal efficacy is held open. Second, *with respect to disclosure* (§4.8): the framework is presented at the level of its organizing principles and a small number of representative examples; the complete operator assignment of the implemented system is characterized through its structure rather than reproduced in full.

The paper proceeds as follows. §2 sets the historical baseline and the revival's operative turn, and locates the work within the study of contemporary esotericism. §3 states the layered epistemology and the "as-if" stance that govern the rest. §4 develops the compositional model. §5 gives worked examples; §6 discusses the relation to attested bind-runes and to the mechanism account; §7 states limitations; §8 outlines future work; §9 concludes.

This is a **conceptual methods paper**: it proposes a design framework and a structural validation criterion for a contemporary constructed practice; it does not report an empirical evaluation of the

framework, nor reconstruct a historically attested runic grammar. Its contribution is conceptual and methodological, illustrated through worked examples and explicit well-formedness checks (§§4.6, 5), not demonstrated by an efficacy study.

The problem motivates three research questions, revisited in §9:

- **RQ1.** How can a contemporary operative rune-stave be represented as a *compositional structure* rather than a thematic collection of symbols?
- **RQ2.** What explicit *well-formedness criteria* distinguish a structurally coherent construction from an arbitrary assemblage?
- **RQ3.** How can such a framework remain *epistemically explicit* about its historical, psychological, and metaphysical claims?

2. Background and Sources

2.1 The historical baseline: what the inscriptions attest

The runic corpus does attest practices that later traditions would develop into bind-runes and runescripts — but as *a few techniques, not a codified system* (MacLeod & Mees 2006). Migration-Period objects carry recurring **charm-words**: `alu` (the most common; its sense is disputed — 'dedication', on the reading of MacLeod & Mees by a North-Etruscan votive parallel, is one of several proposals), `laukaz` ('leek', associated with fertility and potency), `laþu` ('invocation'), and `auja` ('be lucky') (MacLeod & Mees 2006). Two compositional techniques appear "in embryo": rune-**repeats** and rune-**binds**. The Lindholm bone (Scania) bears the line `ek erilaz sawilagaz haiteka` together with an encoded chain of roughly twenty-one runes and the formula `alu` — a *repeat*; the Kylver stone (Gotland, c. 400 CE) bears, beside the oldest *known* complete Elder Futhark row, a "tree-like" figure that MacLeod & Mees leave uninterpreted — a structural antecedent of the *bind* (MacLeod & Mees 2006). The performer is named by the term `erilaz`, traditionally glossed "one skilled in runes and their magic" — but MacLeod & Mees reject this, reading `erilaz` as an ablaut variant of the title "earl," a military-social rank, and explicitly discarding "rune-master" (MacLeod & Mees 2006; Mees 2003).

The boundary this establishes is decisive for the present paper. The inscriptions furnish a *historical root* for bind-runes and rune-repeats, but they furnish **no operative grammar**: no fixed esoteric meaning per rune and no system of layouts. For the Anglo-Saxon material specifically, Page finds the evidence for rune-magic "slight" (Page 1995); the Scandinavian and continental corpus, by contrast, is precisely where MacLeod & Mees locate genuine charm- and amulet-magic. Either way, whatever systematic operative logic a runescript may have is *not* inherited from antiquity; it is a later construction.

2.2 The revival's operative turn

That construction belongs to the modern **runic revival**. From the early twentieth century onward, a succession of authors assigned operative roles and correspondences to the runes and organized them into systems. Two of these must be named with care and kept apart. Guido von List's *Armanen* row (1908) is an **Ariosophic** construction embedded in völkisch ideology, with a documented line to later Nazi rune-appropriation; it is referenced here strictly as history of ideas, not endorsed, and its ideological baggage must not be laundered (Goodrick-Clarke 1992; von Schnurbein 2016). Separately — and ideologically unrelated — Sigurd Agrell's runic number-mysticism (1927), later elaborated into the "Uthark" reordering, is a numerological re-ordering that **mainstream runol-**

ogy rejects and that survives only in modern esoteric practice; it is recorded as such, not as a live scholarly hypothesis (Agrell 1927). Later figures work this same revival vein: the systematized magical practice of Thorsson and the accessible esoteric synthesis of Paxson (Thorsson 1984, 1987; Paxson 2005). It is this revival layer — not the inscriptions — that first treats a rune as bearing a stable *operative function*. The framework developed here is a member of this family: a modern constructive system, continuous with the revival's operative turn, and to be assessed as such rather than as a reconstruction of historical practice.

2.3 Disciplinary location

Because its object is a contemporary constructive practice, the work belongs to the **academic study of Western esotericism**, a field equipped to describe modern esoteric systems rigorously and without either debunking or credulity, on its principle of methodological agnosticism (Hanegraaff 2013); the runic revival in particular has its own dedicated scholarly treatment (von Schnurbein 2016). Situating the framework there, rather than in "rune magic" as a how-to genre, is what allows its claims to be stated as claims *about a practice* and held to ordinary standards of evidence and honesty.

3. Epistemic Framework

The framework is presented within an explicit **layered epistemology** that the larger project applies to every claim it makes. Each assertion is tagged by the kind of thing it is:

- `historical-fact` — confirmed by inscriptions or academic runology;
- `revival-claim` — constructed in the twentieth–twenty-first century (von List, Agrell, Thorsson…), and *not* to be passed off as antiquity;
- `practice-instruction` — a modern practice: what to do;
- `mechanism-evidence` — the psychology of *why* it works;
- `[unverified]` — literal magic or external causation, not proven.

Two rules bind these: reconstructions are not attestations, and revival is not fact; and language such as "the energy of the world" denotes a psychological technique, not a demonstrated transfer of energy.

On this scheme the **central question** — does an operative runescript literally change external reality — is held **open**: neither asserted as a premise nor dismissed. The practitioner (and the analyst) instead runs an experiment and records outcomes honestly. Crucially, holding the question open is *not* a refusal to believe. Belief is itself a working mechanism — expectancy and self-efficacy are documented drivers of motivation and persistence (Bandura 1997; Oettingen 2014) — so one can **invest fully while holding the metaphysics open**. This is the **"as-if" stance**, given a classic philosophical statement by Vaihinger: useful fictions are held *as if* true while known to be constructed (Vaihinger 1924). One may likewise address a rune as a presence without passing that presence off as a proven fact about the world. The project leaves open three **practitioner construals** of what a rune is — an attentional anchor-symbol, an "energy," or an egregore (a collective thought-form) — without adjudicating among them; the "energy" and egregore construals are held in the `revival/[unverified]` layer, not asserted as fact.

Honesty, on this account, lies not in withholding belief but in two disciplines: keeping a log of both hits and misses (the standing guard against confirmation and survivorship bias), and never confusing the layers. This governs everything in §4. The assignment of operative roles to runes (§4.3) is

`revival-claim/practice`, not historical fact; the runtime account (§4.7) is `mechanism-evidence`; and the question of efficacy beyond the practitioner remains `[unverified]`. The model can thus be adopted in full as a constructive, psychologically grounded discipline, with its metaphysics left honestly open.

---

4. The Compositional Model

4.0 Design requirements

Before the model, we state the requirements it is built to satisfy; each later choice answers to one of these.

- **Compositionality** — complex constructions are built from a small operator set over typed parts.
- **Directionality** — the structure is a directed graph with explicit input and output, not a symmetric ornament.
- **Explicit initial and target states** — every construction names where it starts and what counts as done.
- **Inspectability** — the structure can be checked against rules without reference to efficacy.
- **No unused elements** — every element plays a role in reaching the target.
- **Bounded execution** — every loop has a declared termination, abort, or reassessment condition.
- **Separation of syntax from historical claims** — the grammar carries no claim of historical attestation.
- **Mapping consistency** — one versioned operator mapping is applied uniformly.
- **Versionability** — every output is relative to a declared mapping version.

The grammar of §§4.3–4.6 is *one* way to meet these requirements, not the only one.

4.1 Overview: from intention to traceable program

The framework treats the construction of a runescript as a **compilation** — in the computing sense: the process by which a *compiler* translates a high-level description into an executable form. A structured statement of intention is the *source*, and a directed graph of runes — drawn as a single composite figure, the *stave* — is the *compiled output*. Between them sits an intermediate structure, an **abstract syntax tree** (AST: the structured, hierarchical representation a compiler builds from its source), whose well-formedness can be checked independently of any claim about efficacy (Figure 1).

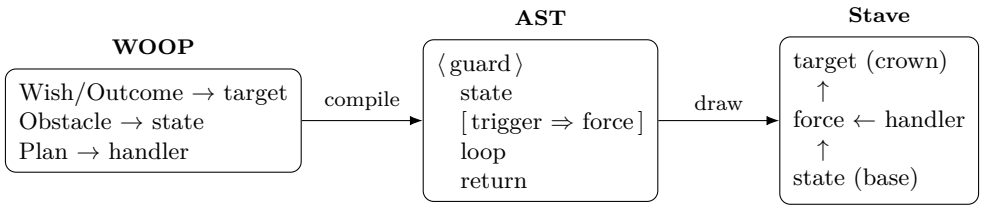
Definition (program). For the purposes of this paper, a *program* is a finite, directed, compositional representation that specifies (a) an initial state, (b) a target state, (c) one or more state-transition operations, (d) the conditions under which those operations are invoked, and (e) a termination or completion condition. The term is used at the level of an *executable specification*, not as a claim that the stave constitutes machine-executable software. Correspondingly, *compilation* is used in a constrained design sense — the systematic transformation of a structured intention into an intermediate representation and then into a drawable directed figure — not as a deterministic compiler in the computer-science sense; the framework does not (yet) specify such an algorithm. *Abstract syntax tree* is likewise used loosely, for an AST-like intermediate representation, and *type*

system for an informal operator-role taxonomy; neither is given a formal grammar or typing calculus in the body (an abstract syntax is sketched in Appendix A).

This inverts the conventional procedure. In a widespread mode of revival practice, a practitioner selects runes *thematically* — by the meaning each rune is felt to carry — and arranges the chosen runes *symmetrically*, so that the result is balanced and pleasing to the eye (Thorsson 1984; Paxson 2005). The arrangement is governed by an aesthetic of symmetry rather than by any executable logic; a stave so produced is, in the terms developed here, a *static charm*. The model below replaces thematic selection with selection *by operational role*, and symmetric arrangement with a *directed* structure in which every stroke is present because the structure of the intention placed it there.

We develop the model in five moves: the source (§4.2), runes as typed operators (§4.3), orientation and inversion (§4.4), the compositional operators (§4.5), and the compilation and its well-formedness criterion (§4.6). §4.7 specifies the runtime model and its epistemic status; §4.8 states the level of disclosure adopted here.

Figure 1. The compilation pipeline. A structured intention (WOOP) is compiled into an abstract syntax tree, which is then drawn directionally as the *stave* — a directed graph. Shown at the level of operator *classes*; the specific rune assigned to each slot is part of the implemented system (§4.8).



4.2 The source: a structured statement of intention

The model does not take a bare wish as input. It takes a *structured* intention, formalized with the **WOOP** schema — Wish, Outcome, Obstacle, Plan — drawn from the psychology of goal pursuit (Oettingen 2014; Gollwitzer 1999). WOOP supplies four components the compilation requires: a goal (Wish/Outcome), an initial condition including what stands in the way (Obstacle), and an if-then plan that specifies which action follows which cue (Plan). The Plan component is, in the psychological literature, an *implementation intention* — an "if situation X, then response Y" linkage shown to improve follow-through (Gollwitzer 1999). The conditional-handler construct (§4.5) is **directly inspired by** implementation intentions; it is not claimed to be *identical* to them, since the framework wraps that if-then core in a larger structure — graph, loops, guard, `return`, sub-routines — absent from the bare implementation-intention form.

The choice of a structured intention over a free-text one is what makes the rest of the model possible: it gives the compilation a *target* to advance toward and a *state* to advance from, so that "advancement" is something that can be checked rather than merely asserted. (We use *structures* deliberately: WOOP provides a standardized elicitation template for the intention, not a formal specification language; and Wish and Outcome are kept as distinct components, jointly fixing the goal, rather than silently merged.)

State, target, and "advance." A *state* is a structured description of the practitioner-relevant conditions assumed to bear on progress toward the target; it may contain project status, internal obstacles, available resources, and observable trigger conditions, which are

best kept distinct — e.g. `state.work_status = stalled`; `obstacle = avoidance`; `trigger = perceived_task_overload`; `target: work_status = shipped`. The *target* is an **observable predicate**, not a mood: *shipped* is binary; *established* requires declared indicators; *decided* requires a recorded decision. Correspondingly, an operation is said to **advance** the state only *relative to an explicitly stated rationale or progress measure*; absent such a measure, any desired interpretation could be declared an advancement — exactly the interpretive-adequacy failure §4.6 guards against.

4.3 Runes as typed operators

The central move of the framework is to treat each rune not as a theme but as a **typed operator** — an element with a grammatical role in a small type system. The roles partition into classes that function like parts of speech:

- **target** — the result or output the program is for (a result-noun);
- **force** — an active operation that moves the state toward the target (a verb);
- **control** — flow control: trigger, condition, loop, branch;
- **modifier** — a qualifier attached to another operator (an adjective/adverb);
- **guard** — a protective wrapper or scope around a sub-program (a `try`/scope);
- **connector** — a binary operator joining two elements;
- **state** — data, resource, or the runtime self-reference.

A rune is then selected for a position *because its operational class fits the slot the compiled structure requires*, not because its thematic meaning resonates with the wish. Two consequences follow. First, many runes are **polysemous** — they carry more than one class (as a word may be both noun and verb), and the operative role is fixed by the compilation context rather than by the rune in isolation. This polysemy is **constrained, not open-ended**: each rune is assigned a *finite, predeclared* set of permitted roles in a versioned mapping (§4.8), and compilation may select only among those predeclared roles — it does not invent a role ad hoc for a given intention. That constraint is what keeps the typing contentful rather than arbitrary, and the method consistent across constructions. Second, the assignment of classes to runes is, on the framework's own terms, a **modern constructed mapping** (a `revival-claim`/`practice` layer in the sense of §3) — chosen by the system's authors, informed by but not derived from the attested rune names. It is not presented as historical fact.

By way of illustration — and only by way of illustration; the complete assignment is withheld per §4.8 — some role assignments align readily with a rune's received sense. The rune of the self most naturally fills the **state** role of the *runtime observer*, the agent whose sustained attention "runs" the program; the sun-rune most naturally fills a **modifier** role as an *amplifier* (×n) that intensifies the operator it attaches to. In these cases we found it natural to assign the role we did, given the rune's received sense; but we do not claim the assignment tracks conventional meaning in general — much of it does not, and the operative role is often orthogonal to a rune's received sense. The higher-level architecture of the framework does not require one uniquely correct mapping, although any concrete implementation and its outputs necessarily depend on a fully specified mapping (§4.8); what is load-bearing at the level of method is that *some* consistent total assignment exists and is applied uniformly.

4.4 Orientation and inversion

Orientation acts as an **operator modifier**. Within this system, a 180° rotation (the *merkstave* orientation — a term from rune divination for an inverted rune) marks a **designated counter-**

operation: where an operation has an explicitly paired reversal — an undoing, release, or withdrawal — its rotated glyph denotes that paired operation. This roughly doubles the effective operator set without enlarging the glyph inventory.

Four points of discipline attach to this. **(a) Graphic convention:** the rotation is a notational mark internal to this system, nothing more. **(b) Semantic convention:** the counter-operation is **author-defined for each reversible operator**, fixed structurally (what it would take to undo that operation) rather than imported from divination; it is explicitly *not* the divinatory "reversed-meaning"/shadow reading. **(c) Historical disclaimer:** this convention is **not** claimed as a historically attested general semantics of reversed runes. **(d) Partiality:** the pairing is **partial** — not every operation has a well-defined counter-operation. A subset of primitives (those assigned to rotationally invariant glyphs) are treated as **non-invertible constants/keywords** — for instance, a cyclic/iteration constant and a source constant; these have no counter-operation (one does not "reverse" an iteration). The remainder are reversible operations, each paired with its undoing where such a pairing has been defined (e.g. acquire/release, engage/withdraw). This alignment of non-invertible primitives with rotationally invariant glyphs is a **design property of the present mapping**, not an independently derived historical or mathematical result.

4.5 The compositional operators

A small, fixed set of operators composes typed runes into a program. Stated as an abstract syntax — independent of how any particular rune is drawn:

- **sequence**, `A` $\rightarrow$ `B` $\rightarrow$ `C`: a pipeline; the execution flow runs through A, then B, then C.
- **binding**, `A` $\cdot$ `B`: a **binary dependency relation** in which `B` specifies the channel, means, or constrained context through which `A` is applied — "A acts *through* B." Drawn, a binding produces a bind-rune proper. (For instance, a force applied *through* a guard is constrained to that guard's scope.)
- **modification**, `A^m`: modifier `m` attaches to operator `A`.
- **conditional handler**, `[ trigger` $\Rightarrow$ `action ]`: an if-then handler — the program "hangs running," and a real-world cue fires the bound action. (This is the WOOP Plan / implementation intention of §4.2, now as syntax.)
- **bounded iteration**, `loop( … ) until reached(target)`: a control cycle — observe, correct, repeat — that terminates on goal attainment. To be genuinely *bounded*, a loop must declare a stopping rule beyond the target alone — a **maximum review count, a time horizon, or an explicit reassessment condition** — since an unreachable target would otherwise yield a non-terminating loop.
- **guard**, $\langle$ `A` $\rangle$: a **scope/boundary annotation** indicating that the enclosed sub-program is subject to an explicit safety or boundary constraint. (It is a scope marker, not exception handling; no `try`/catch semantics are imported.)
- **structure markers**: `state(S)` names the initial condition (input). `target(G)` names the **desired terminal state**, expressed as an observable predicate. `return R` names the **observable, recordable, or materially instantiated completion evidence** produced when the termination condition is met. The distinction is load-bearing: `target` is the *condition to be reached*, while `return` is the *recorded proof that it was* — e.g. target: *manuscript completed*; return: *the submitted manuscript, a DOI, the saved final file.* A construction with no defined `return` is treated as ill-formed; requiring observable completion is what prevents the figure from collapsing into an open-ended charm and is the framework's structural guard against post-hoc success-claiming (§7).

Reusable structure is available through a **sub-routine** primitive — a named, encapsulated sub-program with **explicit inputs and outputs**, **non-recursive**, invocable (including shared invocation in the drawn stave) from more than one place — which lets complex intentions be factored rather than flattened.

Glyph and node. In the rendered stave, the mapping from drawn strokes to program nodes is not guaranteed to be one-to-one: a single stroke may participate in more than one bound rune, and overlapping lines can make a finished glyph **ambiguous to parse** without its accompanying specification. The stave is a *rendering* of the intermediate representation; the representation, not the drawing, carries the formal structure. The framework therefore specifies **forward construction** (intention → representation → stave) and does **not** guarantee lossless **reverse parsing** (stave → representation) from the finished composite glyph alone (see §7).

Figure 2. Node taxonomy. The node/operator kinds of the intermediate representation (§§4.3, 4.5). Shown by class; no rune assignments.

Node / operator	Role
⟨ guard ⟩	scope/boundary around the sub-program
state	initial condition (input)
target	desired terminal state (observable predicate)
return	observable completion evidence
[trigger ⇒ action]	conditional handler (if–then)
action	force | sequence | binding | modified | branch | call
branch	one condition → two or more paths
loop	repeat until target reached, or a declared bound

4.6 Compilation and the traceability criterion

Given a structured intention (§4.2), the typed operators (§4.3–4.4), and the compositional operators (§4.5), compilation produces an **AST-like intermediate representation**: a guarded scope enclosing a `state`, a `target`, one or more conditional handlers whose actions are *intended to* advance the state, a control loop, and a mandatory `return`. The stave is then this structure *drawn directionally* — input (the obstacle) at the base, flow upward through the operations, the handler branch to the side, the loop as a return-edge, the target at the crown, the whole within its guard. It is a **directed graph rather than a symmetric ornament**; the framework imposes no symmetry constraint, so asymmetry may arise where branches encode distinct functions, without being required.

Validation proceeds at **two levels**, which must not be conflated.

A. Structural well-formedness — relatively objective, checkable against the representation:

- a `state` and a `target` (as an observable predicate) are explicitly defined;
- at least one path exists from `state` to `target` (reachability);
- every trigger is defined, and every triggered action has an outgoing continuation or a terminal state;
- every node is connected to the executable graph (no disconnected or unused elements);
- operator arity is respected;
- every loop has an explicit termination, abort, or reassessment condition;
- a `return` / completion condition is defined.

B. Interpretive adequacy — requiring author or practitioner judgment:

- each state-transition carries an *articulated rationale*, not merely a thematic association;
- each operator is judged relevant to the transition it is meant to effect;
- `target` and `return` are operationally specified (observable), not vague;
- no element is justified by thematic resonance alone.

Only level A is checkable in a relatively objective way. We do **not** claim that "each operation actually moves the state toward the target" is fully formally verifiable; what can be checked is whether each transition carries an explicit, articulated rationale — the operation's *claimed* effect — not whether that effect obtains in the world.

This gives the construction explicit, **refutable** structural criteria: a candidate stave can be rejected — independently of any belief about magical efficacy — if it lacks a target, contains a node with no path to it, or has a handler with no trigger. The point bears emphasis: **structural failure does not empirically falsify the metaphysical or behavioral efficacy of a runescript; it shows only that the artifact fails the framework's own construction rules.** This is a weak but genuine notion of correctness, and it is what distinguishes a compiled runescript from a thematically assembled charm.

4.7 The runtime model and its epistemic status

The model includes an explicit account of *execution*. To keep its epistemic commitments clear, we separate five levels.

Representational function. The stave externalizes and compresses the intention structure into a single inspectable figure.

Behavioral execution. The **runtime is the practitioner over time**: across weeks, the practitioner rehearses the structure and responds to its encoded cues. Control flow is the if-then plan as a real behavioral cue-response; sustained attention is what keeps the plan active.

Review cycle. Periodic, honest review (the loop) updates either the behavior or the plan, and records both hits and misses.

Documented mechanisms. The components embedded in this runtime have empirical support *as components*: implementation intentions (Gollwitzer 1999), mental contrasting (Oettingen 2014), selective attention to goal-relevant cues (Moskowitz 2002), expectancy effects on motivation (Wigfield & Eccles 2000), the structuring and anxiety-regulating effects of ritual (Hobson et al. 2018), and self-efficacy (Bandura 1997).

Unverified extension. Any external causal effect *not* mediated by the practitioner's own attention and behavior lies outside the evidential claims of this model and is held as `[unverified]` (§3).

A scope note that the rest of the paper depends on: the psychological literature supports the *components* embedded in the method, **not** the *incremental* efficacy of representing those components as a rune-stave rather than, say, as a written if-then plan. The model is offered as a constructive, psychologically grounded discipline — with no claim of literal external causation, and no claim that the stave form adds efficacy beyond its components.

4.8 Level of disclosure

This paper characterizes the framework at the level of its **organizing principles and representative examples**. The operator typology (§4.3), the inversion principle and the constant/operation

split (§4.4), the compositional operators (§4.5), and the two-level validation criterion (§4.6) are the contribution and are given in full. The system's **complete operator assignment** — the total mapping from each rune (and each orientation) to a specific operation — is an evolving implemented artifact; it is summarized here through its structure and a small number of representative mappings rather than reproduced as a table.

Two consequences must be stated honestly. First, **mapping versioning:** because the assignment evolves, every compiled stave is *relative to a declared mapping version* — a construction references the version under which it was produced, validation is performed against that version, and two staves produced under different versions are not assumed to be semantically equivalent. Second, **the limits of reproducibility:** the framework's *higher-level architecture* does not require one uniquely correct assignment — the contribution is the method of construction, which any sufficiently specified assignment instantiates — but any concrete output *does* depend on a fully specified mapping. The method is therefore **conceptually reproducible** (another researcher can build a system on these principles), while the specific input-to-stave outputs of the implemented system are **not implementation-reproducible** from this paper alone.

---

5. Worked Examples

The examples below are given in the **abstract notation** of §4.5: the *compiled source* is the formal representation, and an accompanying prose description conveys the directed-graph layout. Concrete rune assignments and any rendered per-example glyph belong to the implemented system and are out of scope here (per §4.8); the shared shape these examples instantiate is the compilation pipeline of Figure 1. Operator slots are written by class (`force`, `guard`, `control`, …); where a representative mapping has already been disclosed (§4.3) it is named. We adopt the structure markers of §4.5: `state(S)`, `target(G)`, the guard ⟨ … ⟩, the handler `[ trigger ⇒ action ]`, the loop `loop( … ) until reached(G)`, and the mandatory `return R`.

5.1 A single-handler runescript

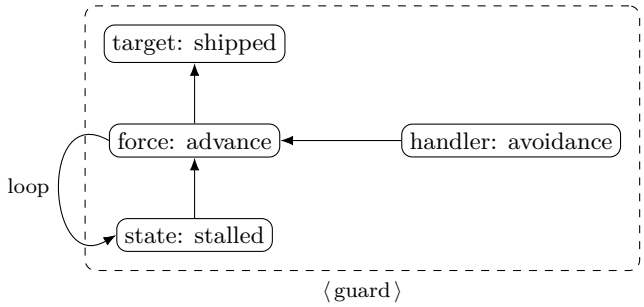
Intention (WOOP). *Wish:* finish a stalled piece of work. *Outcome:* the work shipped. *Obstacle (internal):* avoidance when the task feels large. *Plan:* when avoidance arises, take the single next concrete step.

Compiled source.

```
program finish_work:
  < guard >
    state  = ( stalled, avoidance )
    target = ( shipped )
    [ trigger: avoidance  =>  force: advance(next_step) ]
    loop( observe; correct ) until reached( shipped )
    return  store( shipped )
```

Intermediate representation & glyph. Input (*avoidance*) at the base; a single force-operator carries flow upward; the handler is a side-branch keyed to the trigger; the loop is a review return-edge; the target (*shipped*) crowns the figure, inside the guard.

Figure 3. The §5.1 construction as an abstract stave — a directed graph of operator-class nodes, with no rune assignments (per §4.8).



Validation.

Check (§4.6)	Level	Result
`state` and observable `target` defined	structural	✓ — `stalled` → `shipped` (binary)
path `state` → `target` exists	structural	✓ — via `force: advance`
triggers defined; no inert nodes	structural	✓ — trigger `avoidance`
loop has a bound	structural	✓ — `reached(shipped)` + periodic review
`return` / completion evidence	structural	✓ — the shipped file
articulated, non-thematic transition rationale	interpretive	✓ — the next concrete step reduces avoidance

Epistemic classification. The construction is `practice`; the runtime mechanism (an implementation intention) is `mechanism-evidence`; no `[unverified]` claim is made. **Both levels pass.**

5.2 A multi-handler stave with a sub-routine

Intention (WOOP). *Wish:* steady the first month of a new undertaking. *Outcome:* the undertaking established. *Obstacles:* two recurrent ones — a drop in energy, and conflict that derails the day. *Plan:* a distinct response to each, over a shared restorative routine.

Compiled source.

```
program establish:
  < guard >
    state  = ( new_undertaking, low_energy?, conflict? )
    target = ( established )

    def restore():                          # reusable sub-routine
        amplify( recover )^(sun)            # sun-rune as amplifier (×n), §4.3
        return state'

    [ trigger: low_energy  =>  force: restore() * advance ]
    [ trigger: conflict    =>  control: branch( de-escalate , else step_back ) * restore() ]
    loop( weekly: observe; correct ) until reached( established )
```

```
    return  store( established )
```

Glyph. Two handler-branches leave the trunk at distinct trigger-points; the shared `restore()` sub-routine is drawn once and invoked from both; a branch forks the conflict-handler; the weekly loop is the long return-edge; the target crowns the trunk under the guard.

Validation.

Check (§4.6)	Level	Result
`state` + observable `target`	structural	✓ — `established` (declared indicators)
reachability; no inert or unused arms	structural	✓ — both branch arms lead forward
triggers defined and disjoint	structural	✓ — `low_energy`, `conflict`
sub-routine I/O explicit; non-recursive	structural	✓ — `restore()` returns an updated state
loop bound; `return` / completion evidence	structural	✓ — weekly review + `reached(established)`
each transition has a non-thematic rationale	interpretive	✓

Both levels pass.

5.3 An inversion-as-counter case

Intention (WOOP). *Wish:* release a fixation that blocks a decision. *Outcome:* the decision made freely. *Obstacle:* over-control — gripping the outcome. *Plan:* when the grip tightens, deliberately let go of control.

Compiled source.

```
program decide_free:
  < guard >
    state  = ( fixated, over_control )
    target = ( decided )
    [ trigger: grip  =>  force_inv: release(control) ]    # _inv = inverse operator, §4.4
    loop( observe; correct ) until reached( decided )
    return  store( decided )
```

Here the operative move is an **inverted** operator (§4.4): rather than applying a force to push toward the target, the program applies the *inverse* of a control-force — a deliberate release. This is one of the reversible operations whose inversion is grounded in a documented practice (letting go of control / non-attachment to outcome), not invented for the figure.

Validation.

Check (§4.6)	Level	Result
`state` + observable `target`	structural	✓ — `decided` (a recorded decision)

Check (§4.6)	Level	Result
path `state` → `target`	structural	✓ — the release unblocks the path
trigger defined; no inert nodes	structural	✓ — trigger `grip`
counter-operation *defined* for this operator	structural	✓ — release is the operator's declared pairing (§4.4)
loop bound; `return` present	structural	✓
rationale grounded in documented practice, not theme	interpretive	✓ — letting go of control (§4.7)

Both levels pass.

5.4 What the examples show

Across the three cases the same checkable structure recurs: a guarded scope; a typed `state` and observable `target`; one or more triggered handlers whose operators (forward or inverted) advance the state; a bounded loop; and a mandatory `return`. Each passes both levels of validation (§4.6) before any question of efficacy is raised. The negative examples below show the other half of the test — the criterion rejecting figures that fail.

5.5 Negative examples

The validation criterion earns its keep only if it *rejects* things. Two failure modes, plus a note on symmetry.

(A) Structurally ill-formed. A figure with no `target`, a decorative rune connected to nothing, a handler whose trigger is never defined, and a loop with no stopping rule:

```
program drift:
  < guard >
    state  = ( restless )
    [ trigger: --    =>  force: intensify ]    # undefined trigger
    ornament                                  # disconnected node
    loop( ... )                                 # no termination/bound
    # no target, no return
```

This fails structural well-formedness on four counts: no `target` (and so no path to one), an undefined trigger, a disconnected node, and an unbounded loop. **Rejected at level A** — before any question of meaning.

(B) Structurally valid but interpretively weak. A figure can pass every structural check and still fail level B:

```
program prosper:
  < guard >
    state  = ( wants_wealth )
    target = ( wealthy )                        # observable? barely
    [ trigger: morning  =>  force: wealth-themed-rune ]   # chosen by theme
```

```
loop( weekly ) until reached( wealthy )
return  store( wealthy )
```

The graph is connected, the target is reachable *syntactically*, and a `return` is present — yet the action's only justification is that a rune *means* wealth (a thematic association), the target is not operationally specified, and no articulated rationale links the morning trigger to any change in the obstacle. **Passes A, fails B.** This is exactly the case the two-level split exists to separate: structural coherence does not confer interpretive adequacy.

Asymmetry is not the criterion. These cases also show why *asymmetry* cannot substitute for correctness. Example (B) could be drawn perfectly asymmetrically and remain incoherent; conversely, a simple symmetric figure — two parallel handlers of identical structure answering symmetric obstacles — can be fully well-formed and interpretively adequate. Symmetry and asymmetry are by-products of the branch structure, not tests of validity.

Figure 4. Validation examples (§4.6) — a well-formed construction and three that fail *structural* well-formedness.

(a) well-formed	**(b) disconnected**	**(c) undef. trigger**	**(d) unbounded**
state ↓ [trig ⇒ force] ↓ target ↓ return	state ornament• ↓ [trig ⇒ force] ↓ target ↓ return	state ↓ [? ⇒ force] (trigger undef.) target ↓ return	state ↓ [trig ⇒ force] ↓ loop... (no bound) (and no target)
✓ both levels	× unused node	× undef. trigger	× unbounded; no target

6. Discussion

Continuity with, and departure from, attested bind-runes. The framework's `binding` operator (`A · B`, §4.5) is a **modern formal analogue** of the rune-bind technique attested in the inscriptions, and its repetition construct is **conceptually comparable to** attested rune repetition (§2.1) (MacLeod & Mees 2006); its control-flow semantics, by contrast, are entirely modern. We claim resemblance and modern extension, not a documented genealogy. What the framework adds is precisely what the inscriptions lack: an *operative grammar* over those techniques and a *traceability criterion* for the result. The honest statement of the relationship is therefore neither "this is how the ancients worked" nor "this is unprecedented," but: a historically attested *technique* (binding, repetition) has here been extended by a modern *grammar* (typed operators, control flow, a well-formedness check). The technique is old; the grammar is twenty-first-century.

Why the "program" framing changes the practice. Reframing construction as compilation has consequences that are independent of metaphysics. It forces the practitioner to specify a `state`, a `target`, and a materialized `return`, and to justify every rune by the operational role it discharges — which in turn makes **unused or structurally inert elements** visible ("dead runes" in the framework's shorthand): decorative or thematically pleasing additions that advance nothing are now detectable and removable. It replaces an aesthetic acceptance criterion (is the figure balanced?) with a structural one (does it trace to its goal?). And it aligns the artifact with documented goal-pursuit psychology: the WOOP/implementation-intention structure that improves follow-through (Oettingen 2014; Gollwitzer 1999) is the very structure the figure is required to encode. The discipline of building a *traceable* figure is, on this reading, the discipline of building a

good implementation intention with a ritual anchor.

Mechanism versus literal efficacy. The program framing is deliberately neutral on the open question of §3. Whether or not literal external causation obtains, a well-constructed runescript is a fully specified cue-response plan carried by sustained attention and ritual — and so has documented behavioral purchase on the practitioner regardless. The "program, not talisman" thesis thus retains force even on the most deflationary reading: the gain is in the *construction discipline*, not in any assumed action at a distance.

A note on method. Bringing the vocabulary of formal grammars and instruction sets to an operative symbol-system is, to our knowledge, an unusual move within the study of the runic revival; we offer it less as a final theory than as a *template* — a way to make any operative symbol-practice explicit enough to be checked, taught, and honestly bounded.

Mapping-independent vs mapping-dependent contributions. It helps to separate what the paper establishes from what any *implementation* supplies.

Mapping-independent (this paper)	Mapping-dependent (an implementation)
the need for `state`, `target`, `return`	which rune implements which role
the compositional operators and their arity	which reversal denotes which counter-operation
graph reachability / connectivity checks	the visual shape of the compiled stave
trigger definition and bounded loops	the semantic adequacy of a chosen rune
the two-level validation criterion	the specific input-to-stave output

The left column is the contribution; the right depends on a fully specified, versioned mapping (§4.8) and is not recoverable from this paper alone.

Threats to validity. Two threats deserve explicit safeguards.

Authorial flexibility. The chief risk is that compilation judgment could post-hoc fit an operator interpretation to any goal, making the method unfalsifiable in practice. Safeguards: a **fixed, versioned mapping** with **predeclared permitted roles** (§§4.3, 4.8); a **documented compilation rationale** for each transition; preservation of the original source intention; **no post-outcome remapping**; and, where feasible, **independent review** of a construction.

Outcome reinterpretation. If the target is vague, almost any result can be declared a success after the fact. Safeguards: an **observable target predicate**, a **predeclared completion condition**, a **logged deadline or time horizon**, and explicit **failed / partial / success** states (§4.5).

Logging protocol. Both safeguards are operationalized by a minimal log. **Before use:** mapping version; date; the exact wish/outcome; the observable target; the trigger; the planned actions; the review schedule; the stop condition. **After use:** the outcome; whether it was failed / partial / success; deviations; alternative explanations; and whether the target was redefined. This does not turn the method into an efficacy study; it makes the honesty principle of §3 operational and is the standing guard against confirmation and survivorship bias.

7. Limitations

Several limits should be stated plainly.

1. **A modern construction, not a reconstruction.** The operator assignment is a `revival-claim/practice` (§3); nothing here recovers ancient technique, and no claim of historical operative grammar is made or implied.
2. **No evidence of efficacy beyond documented psychology.** There are no controlled studies of runic operative practice; reported "results" in the esoteric literature are anecdotal and subject to confirmation and survivorship bias. The framework's warrant extends only to its documented psychological mechanisms; efficacy beyond the practitioner remains `[unverified]`.
3. **Traceability is structural, not empirical.** The well-formedness criterion (§4.6) certifies that a figure coheres *as a program* — that it reaches its stated target with no inert elements — not that it produces the targeted outcome in the world. A traceable runescript can still fail in fact.
4. **Under-determination of the mapping.** The framework fixes the *method*, not a unique assignment: different total assignments of operative classes to runes would yield different — and equally traceable — figures. The contribution is therefore a construction discipline, not a canonical correspondence table.
5. **Disclosure trade-off.** Because the complete operator assignment of the implemented system is withheld (§4.8), this paper enables others to build *a* system on these principles but not to replicate *this* system exactly. We regard the principles, examples, and criterion as sufficient to assess and reuse the method, while acknowledging the limit this places on exact reproducibility.
6. **An informal compilation step.** Role selection under polysemy (§4.3) rests on a compilation judgment that this paper characterizes but does not fully formalize.
7. **Forward-only construction.** The framework specifies forward compilation but not guaranteed lossless reverse parsing from the finished composite glyph; a stave generally requires its source or a legend to be recovered unambiguously (§4.5).
8. **No user study.** There is no usability study, no inter-rater reliability study of compilation, no evidence that different compilers produce equivalent structures from the same intention, no test of whether practitioners can correctly trace a result, and no comparison with text-only WOOP plans. These are left to future work (§8).

8. Future work

The framework invites several next steps — none of which is an efficacy proof of magic:

- a complete formal grammar with denotational or operational semantics;
- a versioned public or partial benchmark mapping, to enable implementation-level reproducibility;
- a parser/renderer, and research on reverse-parsability (stave → representation);
- an inter-rater compilation study: do independent compilers produce equivalent structures from the same intention?;
- a traceability/validation scoring rubric;
- a usability study with practitioners (can they correctly trace a result?);
- a comparison with non-runic implementation-intention diagrams, to test whether the rune form adds anything;
- a prospective logging study of constructions and their outcomes.

9. Conclusion

We have reframed the construction of a runescript from a thematic-and-symmetric craft into the compilation of an inspectable program: a structured intention mapped onto a directed graph of runes treated as typed operators, composed by a small grammar, and subjected to a **two-level validation** — structural well-formedness plus interpretive adequacy — under which a candidate figure can fail explicit construction checks, independently of any belief about efficacy. The account is set within an explicit layered epistemology that distinguishes attested history, modern revival construction, documented mechanism, and unverified metaphysics, and adopts an "as-if" stance toward efficacy.

The contribution is the **method**, which any sufficiently specified operator assignment instantiates. Its validation concerns *compositional coherence*, not historical authenticity or external efficacy: the framework turns rune-stave construction into an inspectable design process — conceptually reproducible at the level of its principles, while the specific implemented mapping stays withheld (§4.8). It may offer a transferable analytical template for other contemporary compositional symbol-practices, though such transfer is not demonstrated here.

Declarations

Competing interests. The author is involved in runoscript.com, a project that includes a commercial, paid implementation of the framework described in this paper (an automated rune-stave builder). The framework itself, an associated course, and the supporting knowledge base are published openly and free of charge; revenue derives from the convenience of automated generation, not from access to the method, the sources, or the knowledge. This competing interest is disclosed for transparency; the paper makes no claim of efficacy that would serve to promote the paid service, and (per §4.8) does not reproduce the implementation's proprietary operator mapping.

Author contributions. Anna Berelet: conceptualization, methodology, formalization, and writing.

Use of AI. Large-language-model tools were used to assist with analysis, literature discovery, structural critique, and drafting. This assistance operated over a source corpus and knowledge base assembled and curated by the author; all sources, claims, interpretations, and final wording were reviewed and verified by the author, who takes full responsibility for the content. No AI system is credited as an author.

Data, code, and materials. No empirical dataset was used. The abstract framework, worked examples, and validation criteria are contained in this paper. The implementation-specific rune-to-operator mapping and the compiler artifacts of runoscript.com are not included in this deposit (see §4.8).

Funding. None.

References

Agrell, Sigurd. 1927. *Runornas talmystik och dess antika förebild.* Skrifter utgivna av Vetenskaps-Societeten i Lund 6. Lund: C. W. K. Gleerup.

Bandura, Albert. 1997. *Self-Efficacy: The Exercise of Control.* New York: W. H. Freeman. ISBN 978-0-7167-2626-5.

Barnes, Michael P. 2012. *Runes: A Handbook.* Woodbridge: Boydell Press. ISBN 978-1-84383-778-7.

Dickins, Bruce. 1915. *Runic and Heroic Poems of the Old Teutonic Peoples.* Cambridge: Cambridge University Press.

Düwel, Klaus. 2008. *Runenkunde.* 4th ed. Sammlung Metzler 72. Stuttgart: J. B. Metzler. ISBN 978-3-476-14072-2.

Gollwitzer, Peter M. 1999. "Implementation Intentions: Strong Effects of Simple Plans." *American Psychologist* 54 (7): 493–503. https://doi.org/10.1037/0003-066X.54.7.493.

Goodrick-Clarke, Nicholas. 1992. *The Occult Roots of Nazism: Secret Aryan Cults and Their Influence on Nazi Ideology.* New York: New York University Press. ISBN 978-0-8147-3060-7. (First published 1985, Wellingborough: Aquarian Press, as *The Occult Roots of Nazism: The Ariosophists of Austria and Germany, 1890–1935.*)

Hanegraaff, Wouter J. 2013. *Western Esotericism: A Guide for the Perplexed.* London: Bloomsbury Academic. ISBN 978-1-4411-3646-6.

Hobson, Nicholas M., Juliana Schroeder, Jane L. Risen, Dimitris Xygalatas, and Michael Inzlicht. 2018. "The Psychology of Rituals: An Integrative Review and Process-Based Framework." *Personality and Social Psychology Review* 22 (3): 260–284. https://doi.org/10.1177/1088868317734944.

List, Guido von. 1908. *Das Geheimnis der Runen.* Guido-List-Bücherei 1. Leipzig and Vienna: Guido-von-List-Gesellschaft.

MacLeod, Mindy, and Bernard Mees. 2006. *Runic Amulets and Magic Objects.* Woodbridge: Boydell Press. ISBN 978-1-84383-205-8.

Mees, Bernard. 2003. "Runic erilaR." *NOWELE: North-Western European Language Evolution* 42: 41–68. https://doi.org/10.1075/nowele.42.04mee.

Moskowitz, Gordon B. 2002. "Preconscious Effects of Temporary Goals on Attention." *Journal of Experimental Social Psychology* 38 (4): 397–404. https://doi.org/10.1016/S0022-1031(02)00001-X.

Oettingen, Gabriele. 2014. *Rethinking Positive Thinking: Inside the New Science of Motivation.* New York: Current. ISBN 978-1-61723-023-3.

Page, R. I. 1995. *Runes and Runic Inscriptions: Collected Essays on Anglo-Saxon and Viking Runes.* Edited by David Parsons. Woodbridge: Boydell Press. ISBN 978-0-85115-387-2.

Paxson, Diana L. 2005. *Taking Up the Runes: A Complete Guide to Using Runes in Spells, Rituals, Divination, and Magic.* Boston: Weiser Books. ISBN 978-1-57863-325-8.

Schnurbein, Stefanie von. 2016. *Norse Revival: Transformations of Germanic Neopaganism.* Studies in Critical Research on Religion 5. Leiden: Brill. ISBN 978-90-04-29435-6.

Spurkland, Terje. 2005. *Norwegian Runes and Runic Inscriptions.* Translated by Betsy van der Hoek. Woodbridge: Boydell Press. ISBN 978-1-84383-186-0.

Thorsson, Edred. 1984. *Futhark: A Handbook of Rune Magic.* York Beach, ME: Samuel Weiser. ISBN 978-0-87728-548-9.

Thorsson, Edred. 1987. *Runelore: A Handbook of Esoteric Runology.* York Beach, ME: Samuel Weiser. ISBN 978-0-87728-667-7.

Vaihinger, Hans. 1924. *The Philosophy of "As If": A System of the Theoretical, Practical and Religious Fictions of Mankind.* Translated by C. K. Ogden. London: Kegan Paul, Trench, Trubner; New York: Harcourt, Brace. (German original *Die Philosophie des Als Ob*, Berlin: Reuther & Reichard, 1911.)

Wigfield, Allan, and Jacquelynne S. Eccles. 2000. "Expectancy–Value Theory of Achievement Motivation." *Contemporary Educational Psychology* 25 (1): 68–81. https://doi.org/10.1006/ceps.1999.1015.

Appendix A. Operator typology — summary

*This appendix summarizes the type system and the grammar at the level of §4.3–4.5. It deliberately contains **no** assignment of specific runes to operators (per §4.8).*

A.1 Operator-role taxonomy (informal type system, §4.3)

Class	Role in the program	Loose analogy
`target`	the result/output the program is for	result-noun
`force`	an active operation advancing the state	verb
`control`	flow control: trigger, condition, loop, branch	control structure
`modifier`	a qualifier attached to another operator	adjective/adverb
`guard`	a scope/boundary annotation over a sub-program	scope annotation
`connector`	a binary operator joining two elements	binary operator
`state`	data/resource; the runtime self-reference	variable/runtime

Runes are polysemous, but the polysemy is **constrained**: each rune carries a finite, predeclared set of permitted classes in a versioned mapping, and compilation selects only among those (§4.3).

A.2 Compositional operators (the grammar, §4.5)

Notation	Name	Reading
`A → B → C`	sequence	pipeline; flow runs A, then B, then C
`A · B`	binding	"A acts *through* B" (a true stitch — a bind-rune proper)
`A^m`	modification	modifier `m` attaches to `A`
`[ trigger ⇒ action ]`	conditional handler	if-then; a real cue fires the bound action

Notation	Name	Reading
`loop( … ) until reached(G)`	bounded iteration	observe; correct; repeat until the goal
⟨ A ⟩	guard	protective wrapper/scope around `A`
`state(S)` / `target(G)`	structure markers	initial condition / goal
`return R`	result	**mandatory** materialized output

Reusable structure is provided by a named, encapsulated **sub-routine** primitive, invocable in more than one place (non-recursive; explicit inputs and outputs).

A.2.1 Abstract grammar (informal)

The compositional structure can be written as production rules. This fixes the *shape* of a well-formed construction and the combinations the system permits; it is not a complete formal grammar with denotational semantics.

```
Program   ::= Guarded( State, Target, Handler+, Loop?, Return )
Guarded(X)::= < guard > X
Handler   ::= [ Trigger => Action ]
Action    ::= Operator | Sequence | Binding | Modified | Branch | Call
Sequence  ::= Action -> Action ( -> Action )*
Binding   ::= Operator * Operator
Modified  ::= Operator ^ Modifier
Branch    ::= branch( Condition , Action , Action+ )
Call      ::= call( SubRoutine )
Operator  ::= Force | Control | Modifier | Guard | Connector | StateRef
Return    ::= ObservableCompletion
```

Arity.

Construct	Arity
sequence →	n-ary (associative)
binding ·	binary
modification ^	unary attachment (one modifier on one operator)
conditional handler	one trigger + one action
branch	one condition + two or more paths
guard ⟨ ⟩	unary (scope over one sub-program)
loop	unary body + one termination/abort/reassessment condition
sub-routine call	by name; non-recursive; explicit inputs/outputs

A.3 Orientation and counter-operations (§4.4)

A 180° rotation marks a **designated counter-operation**, author-defined per operator. The pairing is **partial**: a subset of primitives — those on rotationally invariant glyphs — are non-invertible constants/keywords (for instance, a cyclic constant and a source constant) with no

counter-operation; the remainder are reversible operations, each paired with its undoing where defined. This is a design property of the present mapping, not a historical or mathematical law.

A.4 Two-level validation (§4.6)

Structural well-formedness (relatively objective): `state` and an observable `target` are defined; a path exists from `state` to `target`; every trigger is defined; every node is connected (no unused elements); operator arity is respected; every loop has an explicit termination/abort/reassessment condition; a `return`/completion condition is defined. **Interpretive adequacy** (judgment): every transition carries an articulated, non-thematic rationale; `target`/`return` are observable; no element is justified by thematic association alone. Structural failure shows only that the artifact breaks the framework's construction rules — it does not bear on efficacy.

A.5 Informal operational semantics (§4.5)

A deliberately informal reading — not a denotational or fully formal operational semantics. Let `S` be the set of practitioner-relevant states.

- A **force** operator denotes a partial state-transition `f : S` $\rightharpoonup$ `S` (it is *claimed* to move the configuration toward the target; whether it does is interpretive, not formal — §4.6).
- A **conditional handler** `[ t` $\Rightarrow$ `a ]` applies action `a` when trigger condition `t` is observed.
- A **modifier** `m` transforms an operator: `m(f)`.
- **sequence** `f` $\rightarrow$ `g` is composition: apply `f`, then `g`.
- **binding** `f` $\cdot$ `g` is a dependency relation: `f` is applied *through* `g` (g supplies the channel, means, or constrained context — §4.5).
- a **guard** $\langle$ `X` $\rangle$ is a scope annotation over `X`; it is not itself a state transformer.
- the **target** is a predicate `G : S` $\rightarrow$ `{true, false}`; the program *completes* when `G(s)` holds.
- a **loop** repeats its body until `G(s)` holds **or** a declared safety bound (iteration count, time horizon, or reassessment condition) is reached.
- `return R` records the observable completion evidence once `G` holds.

"Informal" is advised: the framework specifies the shape of execution and the conditions for completion, not a calculus for proving properties of programs.

Appendix B. Validation checklist

Structural checks.

- ☐ Initial `state` is explicitly defined.
- ☐ `target` is expressed as an observable predicate or completion condition.
- ☐ At least one path exists from `state` to `target`.
- ☐ Every trigger is defined.
- ☐ Every triggered action has an outgoing continuation or a terminal state.
- ☐ Every node is connected to the executable graph.
- ☐ Operator arity is valid.
- ☐ Every loop has a termination, abort, or reassessment condition.
- ☐ `return` / completion evidence is defined.
- ☐ Mapping version is recorded.
- ☐ No visual element lacks a corresponding node in the intermediate representation.

Interpretive checks.

☐ Every operator has a written transition rationale.
☐ The rationale is operational, not merely thematic.
☐ Role selection is permitted by the declared mapping.
☐ Alternative interpretations are recorded where material.
☐ Success criteria were fixed before outcome observation.
☐ The target was not redefined post hoc.

Epistemic checks.

☐ Historical claims use runological sources.
☐ Revival claims are labeled as modern constructions.
☐ Psychological claims refer only to supported mechanisms.
☐ No evidence for external metaphysical causation is implied.
☐ Metaphorical language is visibly separated from evidential claims.

Appendix C. Example compilation record

A filled record for the construction of §5.1, in the logging format of §6 (placeholders in brackets).

Before use. Mapping version: `m-2026.1`. Date: [recorded]. Wish / Outcome: *finish a stalled piece of work / the work shipped*. Observable target: `work_status = shipped` (binary). Trigger: the felt pull to avoid when the task seems large. Planned actions: on that trigger, take the single next concrete step. Review schedule: weekly. Stop condition: `reached(shipped)`, or six weekly reviews without progress (then reassess).

After use. Outcome: [shipped / partial / failed]. Deviations: […]. Alternative explanations: [deadline pressure, a collaborator's input, …]. Target redefined? [no].

Appendix D. Epistemic claim-labeling scheme

Every claim in this paper, and in its parent project, is tagged by kind (§3):

Tag	Meaning	Example in this paper
`historical-fact`	confirmed by inscriptions or academic runology	charm-words; the Kylver futhark row (§2.1)
`revival-claim`	a twentieth–twenty-first-century construction	the operator mapping (§4.3); merkstave-as-counter-operation (§4.4)
`practice-instruction`	a modern practice — what to do	the construction method (§4)
`mechanism-evidence`	documented psychology of *why* it works	implementation intentions; ritual cognition (§4.7)
`[unverified]`	literal magic / external causation, not proven	efficacy beyond the practitioner (§3, §7)