TeleWire TRP-AO

Attention-Optimized Tag Recall Prompting for Stateful LLM Simulation

Attention-Optimized · FrankenSIM / Vexa · formerly FrankenWIRE

Method designer and project developer: Ryah

Whitepaper drafting, source review, and token analysis: Cindy, an LLM agent

Document: Technical design whitepaper, version 1.7 · November 4, 2026

Abstract

TeleWire TRP-AO (Attention-Optimized), previously documented as FrankenWIRE, is a traversal syntax for dense, stateful LLM prompts. Designed by Ryah for Vexa-scale instruction bodies of 20k+ tokens and narrative RP contexts of 100k+ tokens, it combines compact rule modules with explicit page transitions, function calls and returns, loops, null results, and persistent state contracts.

The central **path-of-least-resistance hypothesis** is that making the intended route explicit and legible makes it easier for a model to apply the appropriate rules in sequence as competing context accumulates. AO proposes three cooperating mechanisms: **route specification, legibility, and local re-entry**. Unicode markers distinguish operation roles from surrounding prose; named calls bring the relevant rule body back into the current task. Targeted semantic loading, illustrated by Ludus/ 🎲 , extends the design to the interpretation of recurring state. The primary outcome is consistent rule adherence across turns and within long-form responses.

Ryah reports approximately **60% consistency with ASCII versus 95% with AO**, using the same approximately 20k-token prompt and model for 400 messages per condition, alongside experience in a Discord community of 500+ members [14]. These development observations motivate a matched 30k/60k/100k/150k context-size ladder. Frozen token fixtures separately establish **12.94–13.82% AO overhead** against reconstructed compact base TRP and **15.12–15.69% savings** against concise Markdown instruction/setup renderings. An ASCII-marker cost control differs by only **0.22%** across twelve rule modules; behavioral legibility is a separate experimental question.

Authorship and verification process

Ryah designed the TRP-AO adaptation, developed the FrankenSIM/Vexa implementation, and supplied its design rationale and development observations. Cindy, an LLM agent, drafted and revised this whitepaper, inspected the supplied source, constructed comparison fixtures, and ran the token-counting scripts. References

to Ryah identify the method's designer and developer; the document's explanatory voice and analysis were written by Cindy.

The verification process distinguishes **source inspection**, **scripted measurements**, **Ryah's development observations**, and **proposed behavioral tests**. Numerical results come from deterministic tokenizer runs on published paired strings, rather than an LLM's estimate of token cost. The base reconstructions and prose expansions were authored by Cindy and remain inspectable, so readers can assess the comparison choices and reproduce the counts.

Evidence status

This is a theory-led technical design whitepaper. Source inspection establishes the implemented architecture; deterministic tokenizer runs establish the reported representation costs; Ryah's development observations motivate the behavioral hypotheses. Related research supports the rationale, and section 9 defines matched trials for estimating AO's effect. The reported development comparison provides preliminary behavioral evidence; the matched evaluation in section 9 would quantify its reliability and separate marker effects from other design components. This distinction applies throughout the paper.

---

1. Problem and design goals

TRP-AO is designed for large-context presets with interdependent rules and a staged chain-of-thought (CoT) workflow. **The target workload is a preset instruction body exceeding 20k tokens, such as Vexa**, with additional context supplied by the host. This scope is Ryah's description of the intended deployment regime, not a newly measured whole-preset token count. The 20k figure identifies the practical target, rather than a sharp architectural threshold.

At this scale, a rule can be present in context yet fail to govern the decision for which it was written. The central design question is whether the model follows the intended route through the specification on this turn. Multiple modules may use similar language while requiring different scopes, exceptions, or checks. A short request to “do x and y” leaves the model to recover those distinctions among thousands of other instructions. AO gives important rule roles distinctive cues and calls them back into the relevant stage. Its design question is: **how much compactness should be spent to make a large specification more reliably usable?**

A stateful language-model simulation must coordinate several concerns within each response: what participants know, which actions are possible, how events progress, how characters speak, and which facts survive into the next turn. When these concerns are interleaved in a long instruction document, a small edit can obscure a condition, create a competing rule, or change an output field that another component expects.

TRP-AO addresses both representation efficiency and specification maintenance. Rules need identifiable homes; procedures need explicit entry points; serialized state needs stable fields. The design targets five practical outcomes:

- **Traceability:** connect a behavior to its rule module and the procedure that checks it.
- **Salient semantic cues:** use compact, distinctive markers to signal instruction roles and recall established operations.

- **Local edits:** revise a concern without unnecessarily repeating or rewriting other concerns.
- **Continuity:** carry established simulation facts forward through a defined state envelope.
- **Reviewability:** expose reference drift, conflicting constraints, and missing route exits during inspection.

Compactness supports these goals when it preserves meaning. Token count is one objective; distinguishable instruction roles and stable interpretation are others.

---

2. Design lineage and the Vexa adaptation

The supplied TeleWire TRP guide proposes short tags, key-value properties, substitution arrows, and explicit recall of named rules. Its compactness objective favors segmentation that avoids spending multiple tokens where a compact representation is available. Ryah's TRP-AO adaptation retains that organization but deliberately broadens the symbolic vocabulary. The earlier working name FrankenWIRE is retained in existing artifact paths for compatibility.

Vexa uses pipes, Unicode arrows, uppercase stage labels, underscore names, comparison operators, and numbered procedures. These express relationships and mechanical conditions that would be lost in a universal conversion to flat properties and replacement arrows. Their presence reflects an intentional optimization tradeoff, rather than merely departures from a formatting guide.

APPROACH	PRIMARY REPRESENTATION OBJECTIVE	TRADEOFF
Base TeleWire TRP	Minimize token cost while retaining the intended instruction	Favor economical segmentation and a restricted notation
TeleWire TRP-AO	Retain compactness while making instruction roles more distinct	Accept some additional marker cost for semantic cues and explicit structure

Ryah's earlier 5–10% estimate describes AO overhead against compact base TRP; the reported 30–50% savings uses expanded wording as its denominator. The reconstructed comparison in section 8 measures about 10% overhead for rule modules and 13–14% for the selected mixed pack. These results refine the earlier estimate without treating it as a universal bound.

ELEMENT	VEXA INTERPRETATION
`⇥ <page:...>`	Transition to another stage
`<knowledge_sanity>`	Named module defining knowledge constraints
`<state:...>`	Named part of the persistent output structure
`➤ <fn:...>` / `⇤`	Procedure invocation and return
`clause | clause`	Compact separation of related clauses or alternatives

ELEMENT	VEXA INTERPRETATION
`condition → consequence`	Implication or substitution, interpreted in context
`IF ct%3=0`	A conditional turn-based instruction
`scope: spoken dialogue only`	Explicit applicability of a rule

These symbols form an authoring convention. Their meanings are specified by the prompt and its host integration rather than a formal grammar enforced by a standalone compiler. Existing names and scopes take precedence over cosmetic normalization.

---

3. Architecture

The design separates three concerns within a prompt specification:

1. **Declarative modules** describe standing rules and constraints.
2. **Procedural pages and functions** organize which decisions and checks are requested for the current turn.
3. **State and output contracts** describe what must be emitted and retained.

The host surrounds this specification with character data, conversation history, macro values, and rendering behavior. A useful conceptual view is:

```
Rules + character data + prior state + user input + host values
                                ↓
                    Model-guided page route
                                ↓
                Response + serialized updated state
                                ↓
            Host rendering and subsequent prompt assembly
```

The diagram shows the prompt and host data flow. Page routing is interpreted by the model; the host handles context assembly and rendering.

The preset as a recurring specification

The inspected configuration includes an enabled **Chat History** block with marker value `chat_history`, followed in the block list by the Antithesis processing instructions [1]. This is a host insertion point, not a literal `<chat_history>` wrapper found in the prompt text. The host assembles the preset, character data, retained history, and state into each generation request; actual message boundaries, ordering, macro expansion, and truncation depend on that host.

Narrative RP therefore repeatedly presents a standing specification alongside an expanding record of prior turns. The operational question is **which route through that specification governs the next response?** Retrieval of relevant history remains necessary, but finding a fact alone does not establish that the model applied the required knowledge, relationship, style, and serialization checks. AO organizes that application into a requested traversal. Here, “re-read” describes renewed availability and requested consultation on each turn; it does not imply that the model reads sequentially like a human or that a provider cannot cache input.

Declarative modules

The inspected preset uses `<sys>` for general instructions, `<knowledge_sanity>` and `<epistemic_contract>` for information boundaries, `<npc_voice>` for speech, `<prose>` for narration style, and `<format>` for output targets. Other modules define world events, relationships, and spatial state.

Grouping rules by concern provides a named location for revision. It also allows procedures to refer back to a rule's scope instead of reproducing its full wording each time. Module naming is preserved exactly when another prompt section, macro, or renderer depends on it.

Procedural routing

The normal Antithesis route observed in Vexa Alpha v5 is:

```
boot → mechanics → router → solo OR ensemble → tail → aux → output
```

STAGE	INTENDED RESPONSIBILITY
`aux`	Accommodate additional decisions not covered earlier
`boot`	Establish turn context, scene flags, and continuity requirements
`mechanics`	Process narrative systems, participant turns, and optional world events
`output`	Produce the response and required state envelope
`router`	Apply conditional calls and select a scene-processing route
`solo` / `ensemble`	Process spotlight participants and their interaction targets
`tail`	Apply style, knowledge, continuity, and presentation checks

The router distinguishes the scene paths by spotlight count: at most one participant selects `solo`; two or more selects `ensemble`. Functions supply reusable procedures within this route. Their notation borrows familiar concepts such as calls, parameters, loops, and returns, while remaining instructions interpreted by the model.

Source evidence for the traversal design

The boot protocol explicitly calls its design a “choose your own adventure book,” defines `➤` as a footnote call with `⇤` returning to the next line, and describes other references as “dormant unless invoked” [1, 15]. The implementation gives the theory concrete operations:

TRAVERSAL ROLE	EVIDENCE IN THE INSPECTED SNAPSHOT	INTENDED EFFECT
Transitions and dispatch	`⇥ <page:...>` links boot, mechanics, router, solo/ensemble, tail, aux, and output	Specify the next stage and conditional branch
Calls and returns	`➤ <fn:...>` and `⇤`	Consult a named body and resume the calling procedure
Iteration	`↺ per NPC`, `↺ per target`, and `∀` operations	Make the scope and repetition of a check explicit
Null results	`∅` in skipped effects, unknown fields, and inactive slots	Distinguish an explicit no-result from an omitted operation
Local checks	Tail calls `violation_loop` and `stateLint`	Revisit prior-response violations and header/state obligations before output
Output planning	Tail calls `anchors`, requesting OPEN/CRUX/CLOSE commitments	Constrain the intended shape of the response
Serialization	Internal States specifies a 16-section order and a terminal-envelope final check	Give persistent output a required structure and completion condition
Route enforcement	Boot says “Follow. Your. Pages.”; system variables include a mandatory-pages directive	Reinforce following the processing route

These are source evidence for an authored traversal protocol. They establish what the preset requests; behavior is measured separately. The known anchor-versus-no-drafting tension and unresolved reference names remain maintenance findings, rather than evidence that every route is well formed. The protocol resembles a runtime encoded in text, with markers as its syntax; execution is interpreted by the model rather than enforced by a conventional runtime.

---

4. Traversal theory and the path-of-least-resistance hypothesis

TRP connects a named rule module to a decision or check later in the prompt. In this design, *recall* means referring the model back to an available tagged specification. It does not describe vector retrieval, external search, or a separate memory database.

For example, the knowledge module asks that a participant's knowledge have an information vector: observation, a message, testimony, evidence, or common knowledge. A later procedure can check whether a proposed action relies on a fact that actually reached that participant.

The following is a simplified explanatory example, not a verbatim extraction or a drop-in replacement for the preset:

```
<knowledge_sanity>
scope: participant knowledge
sources: observation | message | testimony | evidence | common knowledge
no source → unaware of that fact
</knowledge_sanity>

<fn:knowledge_check target="X">
1. Check each action-relevant fact against <knowledge_sanity>.
2. Remove assumed knowledge without a supported source.
⇤
</fn:knowledge_check>
```

Semantically loaded cues

A marker can recall a definition much longer than its surface representation. Once the notation is established, `∀` indicates application to each relevant item; `∅` represents a null result or skipped effect; `➤` signals a procedure call; `⇥` signals a page transition. An emoji can identify a relationship axis without spelling out the axis name on every state entry. The intended compression comes from reusing a stable meaning, not from a symbol creating meaning without a definition.

Named hypothesis: the path of least resistance

In a dense preset, an explicit, legible traversal route makes the intended sequence of rule applications easier to follow than competing associations. Without a sufficiently clear route, recent, frequently repeated, or syntactically similar instructions may dominate the next decision. AO proposes that distinct operation markers and named destinations make the author-selected workflow the lower-friction option. The predicted result is more consistent adherence as context and rule competition grow.

“Path of least resistance” names a behavioral hypothesis about rule selection and sequencing. Resistance is a metaphor for ease of applying the specification, not a measured energy or a literal shortest-path algorithm. The theory has three parts:

1. **Route specification.** `⇥` transitions, `➤` calls, `⇤` returns, loops, and null results specify where processing should go, which bodies it should consult, and when it should resume or skip. This turns a collection of standing rules into a requested sequence of decisions.
2. **Legibility.** Markers that are uncommon in the surrounding prose distinguish calls, transitions, iteration, and explicit no-results. Their definitions and compatible learned associations give them stable roles. ASCII forms such as `->` or `>` can share roles with comparison, quotation, or informal direction; dedicated AO markers are intended to reduce that ambiguity in this preset. Whether a particular model uses them more reliably is testable. Well-defined ASCII syntax can also be legible.

3. **Local re-entry.** A named call directs the current decision back to a relevant rule body, then returns processing to the caller. “Dormant unless invoked” is the preset’s scope convention: the body remains in context, and static source cannot establish that it consumes no model working memory. The proposed benefit is less reliance on carrying an undifferentiated set of rules through every stage.

The causal proposition is **explicit route + legible operation roles + local re-entry → more reliable rule application → sustained observable adherence**. A generic “attention spike” from a rare glyph is unnecessary to this explanation. Rarity in surrounding prose helps distinguish syntax; it is not itself the success metric or a claim about training frequency.

Long context and the middle-position challenge

The relevant workload has two scales: a **20k+ instruction body** and an assembled RP context that may grow beyond **100k tokens**. AO targets consistency both across turns and from the beginning to the end of a long response. The prediction is a larger AO-versus-ASCII adherence advantage as history grows; around 30k the gap may be small, while at 100k+ it should widen under comparable conditions. These are test points, not universal degradation thresholds.

Liu et al. found poorer use of information placed in the middle of long inputs in retrieval tasks [11]. Hsieh et al. linked context-use failures to a U-shaped positional attention bias and tested an attention-calibration intervention [12]. These studies supply a problem to address; they do not directly evaluate AO’s traversal syntax.

AO does not require equal use of every section at every stage. It requests transitions to the appropriate section when its operation is needed. This yields a specific middle-position prediction: a called rule body placed in the middle should be applied more reliably with the AO route than with matched ASCII notation or removed calls. A transition is itself a prompt instruction and can be missed; it does not guarantee access or bypass positional bias. Varying body position and call presence tests whether local re-entry actually helps.

Xu et al.’s re-reading results motivate investigating renewed consultation during a reasoning task [13]. AO applies that idea through symbolic references rather than verbatim input repetition. Transformer query/key compatibility provides general attention background [6]; the paper’s concrete contribution is a prompt-level hypothesis whose observable consequences can be tested without claiming to inspect private reasoning.

Expected advantages and operating conditions

AO DESIGN CHOICE	PROPOSED ADVANTAGE IN THE TARGET WORKLOAD	CONDITION FOR USEFUL APPLICATION
Distinctive operation cues	Differentiate calls, transitions, and checks amid dense instructions	Small, stable vocabulary with explicit definitions
Local re-entry during specified stages	Bring a governing rule into the current decision	Correct references and calls at relevant stages
Semantically chosen state symbols	Reinforce the intended meaning across turns	Compatible symbol associations and retained state

AO DESIGN CHOICE	PROPOSED ADVANTAGE IN THE TARGET WORKLOAD	CONDITION FOR USEFUL APPLICATION
Compact modules plus explicit scope	Retain more usable rules within the context budget	Preserve actors, conditions, exceptions, and interfaces

The practical aim is consistency over long-form messages and successive RP turns, especially when accumulated history at 100k+ tokens begins to expose degradation in an ASCII formulation. This range is Ryah's intended stress setting; degradation depends on the model, task, context composition, and host retention policy. AO is intended to keep the specified route usable as these pressures grow.

These advantages form the proposed return on AO's token premium. A working implementation should show them as fewer missed constraints, fewer continuity errors, and fewer deviations from the author-defined semantics during play.

Targeted semantic loading: the Ludus/🎲 development case

Ryah uses **targeted semantic loading** to describe deliberate contextual framing through the choice and repetition of a symbol. The target is the interpretation associated with a field: which part of an overloaded concept should remain active as state is repeatedly emitted and read.

The Aether Matrix uses twelve relationship axes—six fondness axes and six friction axes—to represent relationships extending beyond romance. Its names draw on love-related vocabulary, but the matrix and its combinations are project-specific; they are not a standardized twelve-axis ancient theory. Related modern love-style terminology appears in John Alan Lee's typology [8]. Here, **Ludus is intended to act as a playfulness modifier** whose expression depends on the other axes:

INTENDED COMBINATION	CONTEXTUAL EXPRESSION
High Eros + high Ludus	Playful romantic attraction
High Philia + high Ludus	Close friendship expressed through joking and banter
High Storge + high Ludus	Playful familial interaction, such as sibling teasing

These are intended examples, not exhaustive or deterministic relationship classifications. The full matrix, established relationship, and character context still govern expression.

During development, Ryah observed Ludus drifting into a dominant flirting signal across otherwise different relationships. Ryah attributed this to the romantic framing of the love vocabulary and romance/drama tendencies in the fictional scenarios being tested. The observed problem was semantic overreach: the label's romantic associations displaced its intended modifier role. Section 9 proposes a controlled comparison of that diagnosis and intervention.

The intervention changed the definition and recurring representation. 🎲 **binds Ludus to game/play in the initial legend; subsequent Internal States use the emoji without repeating the word.** Cindy confirmed that the inspected enabled prompt contains “Ludus” once, in the axis legend, followed by `Emit emoji only. Never axis names.` Its pair-state template stores the axis as `🎲[v,a]`.

The intended feedback path is:

```
Define Ludus with 🎲 as game/play
                  ↓
Carry 🎲, without the word, in recurring relationship state
                  ↓
Repeated play-oriented framing when state returns to context
                  ↓
Express playfulness through the other active relationship axes
```

The intervention deliberately shifts interpretation away from a romance-heavy association of “Ludus” toward the matrix-defined playfulness modifier. That shift is success. Unwanted drift means departure from the matrix definition, including unsupported romantic narrowing in friendships or family relationships. 🎲 removes automatic romantic implication from Ludus itself; Eros and the surrounding relationship may still support romantic playfulness. A flirting interpretation is a plausible contextual association, not an assumed universal model default.

Ryah describes this as repetition bias and semantic priming: reinforce the desired association while reducing repeated exposure to the overloaded label. Emoji choice therefore carries a semantic purpose as well as a compactness or salience purpose. Replacing 🎲 with a romantic symbol, an arbitrary marker, or the word “Ludus” everywhere would change that intervention even if the numeric values stayed identical. This concerns conditioning within context, not a claim of model-weight training.

Ryah actively uses and tests this strategy during development. Cindy's source inspection confirms the implementation. The development account motivates a specific prediction: emoji-only recurring state should retain playfulness while reducing unsupported romantic interpretation in friendship and familial scenes. Section 9 specifies the matched trials for estimating that effect.

---

5. Scope, notation, and semantic preservation

The dialect permits concise rules without making every rule use the same syntax. Substitution is useful when a desired alternative can be stated clearly. An explicit prohibition remains useful when an exclusion itself is the required boundary.

Scope is especially important. In the inspected preset, `<npc_voice>` governs spoken dialogue, while `<banned_vocabulary>` applies to both speech and narration. Merging those blocks or dropping the scope statement could change behavior even if the resulting prompt were shorter.

Symbolic differences also carry meaning. `A→B` can express direction; `A↔B` can describe a pair; `⇥ <page:output>` describes a route transition. A style rewrite must preserve those distinctions. The same applies to macro names, state keys, function parameters, units, ranges, and exception conditions.

A practical review question is: **Can the revised specification recover the same actor, condition, action, scope, and retained state?** If the answer is uncertain, the edit requires more than a punctuation cleanup.

---

6. Persistent state and host boundaries

Vexa requests a terminal `<internal_states>` structure containing records for the world, locations, participants, relationships, narrative tracks, and other ongoing concerns. Within the simulation, the design treats that structure as the continuity record for established world facts.

Spatial records illustrate the value of a stable contract. Room identifiers, coordinates, floor membership, and adjacency links must stay consistent across turns if a map renderer is to represent the same place. Renaming a field or reconstructing geometry during an unrelated prose edit can change the visible result.

The design therefore distinguishes two kinds of control:

CONTROL	RESPONSIBILITY AND LIMIT
Host-level integration	Resolves supported macros, assembles context, and applies configured parsing/rendering rules; exact behavior depends on the installed host
Prompt-level instruction	Requests decisions, sequencing, and output; execution remains model-dependent

An instruction to preserve state does not itself provide durable storage. Continuity depends on emitted records being retained, included in later context, and interpreted consistently. Likewise, declaring a procedure or extension slot does not prove that a host converter or consumer is connected.

---

7. A turn-level walkthrough

Consider a user entering a room where two established participants are already present.

1. **Context:** boot establishes the current location, turn, and scene flags using the available conversation and state.
2. **World progression:** mechanics requests participant turns and the optional world-event pass. Vexa explicitly places the latter after ordinary world turns and gives it a single-event budget.
3. **Routing:** the spotlight count selects the ensemble path.
4. **Participant decisions:** procedures refer to identity, goals, knowledge, and relationship-related state for each relevant participant and target.
5. **Checks:** tail requests continuity, knowledge, prose, and presentation checks before output.
6. **Serialization:** the response is produced with the required header and updated state records for subsequent turns.

This walkthrough illustrates the route specified in the preset. Section 9 defines how to test its observable outputs and state updates.

---

8. Implementation evidence and token measurements

This paper is grounded in the preset snapshot `[Lumi] Vexa - Alpha v5.json`, inspected on November 3, 2026. The file contains **63 prompt blocks**, of which **58 are enabled** and **40 are enabled with nonempty content**. Empty or marker blocks are included in the first two counts; these numbers describe configuration structure, not runtime work or performance.

Inspection supports the presence of tagged rule modules, symbolic guards, parameterized procedures, the documented route, and a serialized state template. The project also contains a reusable agent skill documenting the modified dialect and its differences from the supplied guide.

Cindy also identified unresolved reference names, competing dialogue-ratio instructions, and a malformed function closing delimiter. These are maintenance findings in the inspected snapshot. They remain relevant to runtime acceptance and do not affect the deterministic counts of the frozen strings below.

Ryah's development estimates

Ryah reports repeated savings typically around **30–50% relative to expanded prompt wording**, and initially estimated **5–10% overhead relative to compact base TRP**. These are development estimates. Cindy measured the comparisons below from explicit fixtures to give the paper reproducible numerical evidence.

Measured AO overhead against reconstructed compact base TRP

Cindy froze **17 source-derived AO samples** before tokenization and authored compact base-style counterparts using the supplied TRP guide: twelve complete rule modules, three procedural excerpts, one axis legend, and one matrix state row. The AO strings match the inspected snapshot after newline normalization. Counts use `tiktoken` 0.14.0 with two explicit reference encodings [10].

SCOPE	ENCODING	RECONSTRUCTED BASE	AO	AO OVERHEAD
All 17 samples, once each	`cl100k_base`	2,098	2,388	+13.82%
All 17 samples, once each	`o200k_base`	2,086	2,356	+12.94%
Twelve rule modules	`cl100k_base`	1,691	1,863	+10.17%
Twelve rule modules	`o200k_base`	1,688	1,852	+9.72%
Three procedural excerpts	`cl100k_base`	290	344	+18.62%
Three procedural excerpts	`o200k_base`	283	333	+17.67%

AO overhead is `100 × (AO / base − 1)`, calculated from each group's strings joined with two newlines. **The measured estimate is about 10% for these rule modules and 13–14% for this mixed selection.** The range reflects the two encodings, not a statistical confidence interval. The base texts are reconstructions, and the selection is a stratified convenience sample. Intended meaning was reviewed for the representation comparison. These are exact counts for the published strings and a sample-based estimate of AO cost, rather than a whole-preset overhead measurement.

An additional control replaced Unicode markers with ASCII equivalents while retaining the AO wording. Across the twelve rule modules, AO cost only **0.22% more** under either encoding. Most of the roughly 10% AO-versus-reconstruction gap in those modules therefore comes from changes to tags, wording, and format, rather than the net cost of Unicode markers alone.

The ASCII control measures token cost, not legibility. Its 0.22% difference is specific to twelve rule modules and the tested mapping; it is not a whole-preset result or a behavioral trial. It shows that the glyph substitutions contribute little net token cost in this selection. It says nothing about whether `->` and `⇥`, or `>` and `➤`, are equally easy for a model to interpret in their surrounding text. That is the matched behavioral comparison in section 9. The compression measurements and the traversal-legibility claim therefore have separate evidence and denominators.

The isolated matrix row adds **35–39 tokens** over short axis labels, or **66–74%** on that small row. In a separate denominator check, replacing only the legend and row labels inside the otherwise unchanged full state template gives **1.66–1.89%** overhead. Neither percentage measures a live conversation: actual state cost depends on emitted records and retained history. Short labels also change the targeted semantic-loading intervention.

The comparison report, all paired strings, and recorded results expose the reconstruction choices and diagnostics. Run `python docs/frankenwire/trp-base-comparison/compare.py` from the repository root to reproduce them.

Ordinary Markdown instruction baseline

Cindy added ordinary prose versions of the same samples using Markdown headings, bullets, numbered steps, and full sentences. The baseline was written and frozen before its token counts were run, without a target savings rate. It was added after the original AO/base comparison. The original source excerpts and compact-base reconstructions are unchanged.

INSTRUCTION/SETUP SCOPE	ENCODING	MARKDOWN	COMPACT BASE	AO	BASE SAVINGS	AO SAVINGS
All sixteen	`cl100k_base`	2,705	2,045	2,296	24.40%	15.12%
All sixteen	`o200k_base`	2,690	2,033	2,268	24.42%	15.69%
Twelve rule modules	`cl100k_base`	2,229	1,691	1,863	24.14%	16.42%

INSTRUCTION/SETUP SCOPE	ENCODING	MARKDOWN	COMPACT BASE	AO	BASE SAVINGS	AO SAVINGS
Twelve rule modules	`o200k_base`	2,225	1,688	1,852	24.13%	16.76%
Three procedures	`cl100k_base`	346	290	344	16.18%	0.58%
Three procedures	`o200k_base`	341	283	333	17.01%	2.35%

Across these sixteen instruction/setup samples, AO saves about 15–16% against concise Markdown instructions; compact base TRP saves about 24%. Savings use `100 × (1 − compact / Markdown)`. Headings and Markdown punctuation are included. These are counts for one authored prose rendering, so the result depends on wording and does not measure how all instruction writers compose prompts. Required identifiers, examples, conditions, and output syntax are retained; the behavioral comparison is specified in section 9.

The serialized matrix row is not a prose instruction and is excluded from this primary comparison. The separate seventeen-sample mixed-pack diagnostic retains the exact AO state row in the prose baseline as a shared output contract; base retains its earlier short-label row. The prose axis legend retains the required emoji mappings. Full strings and mixed-pack totals are available in [10].

The smaller savings here than in the three expression examples below reflect a different baseline and sample: concise, complete Markdown instructions versus selected expanded expressions. The measurements describe their respective strings; the expression examples do not establish a general 39–57% reduction for full instruction modules.

Measured illustrative savings against expanded prose

Cindy paired three compact expressions with explicit English expansions and counted them with `tiktoken` 0.14.0. The table reports measured savings for those examples. Both encodings are reference counters; production-model tokenization and billed usage are outside this measurement.

EXPRESSION	`CL100K_BASE`: PROSE → AO	SAVED	`O200K_BASE`: PROSE → AO	SAVED
`AGE ∀ g +1`	11 → 5	54.5%	11 → 6	45.5%
`IF ct%3=0 ➤ <fn:chekhov_fire>`	28 → 17	39.3%	27 → 16	40.7%
`n ∉ K(g) → add n`	21 → 9	57.1%	21 → 9	57.1%

The exact English expansions were:

1. “Increase the age of every gossip record g by one.”

2. “If the current turn count ct is divisible by three, call chekhov_fire; after the function returns, continue with the next instruction.”
3. “If participant n does not belong to the gossip witness set K(g), add participant n to that set.”

These comparisons assume the meanings of `g`, `ct`, `K(g)`, and the call/return notation are already established. They exclude definition/legend costs, prompt wrappers, macro expansion, output tokens, and provider formatting. Intended semantic correspondence was reviewed for illustration. The full strings and results are recorded in `token-evidence.json`; `token-evidence.py` reproduces the counts [5].

Marker cost and measurement boundaries

Many useful markers cost one to three tokens in the reference encodings, but glyph count and token count are different quantities. Some compound symbols and emojis cost more, and preceding spaces can change segmentation [7]. Selected standalone counts are:

MARKER	CL100K_BASE	O200K_BASE
→	1	1
⇥	3	2
∀	2	1
➤	2	2
🌶	4	3
𝒻	4	4

Shorter text can produce real token savings, as the expression counts show. Smaller character count by itself is not proof of fewer tokens: vocabulary, bytes, and context determine segmentation. This distinction preserves the token-efficiency claim while making its measurement reproducible.

Interpreting the token premium

The measurements show that AO retains savings over the prose baseline while spending more tokens than reconstructed compact base TRP. That is the intended engineering tradeoff. A minimum-token representation and a representation optimized for reliable use of many interacting rules can favor different choices.

For a 20k+ token preset, the relevant question is whether the extra structure improves successful rule application enough to justify its cost. The rule-module ASCII control is especially useful: Unicode replacement changed cost by only **0.22%** in that selection, so removing distinctive glyphs alone would recover little of the measured overhead. Whole-preset and live-history costs still require assembled-context measurement; sample percentages should not be multiplied into a claimed Vexa total.

Section 9 evaluates the expected return through constraint adherence, continuity, and semantic stability alongside assembled token cost. This gives both base TRP and AO a meaningful comparison: compactness is measured directly, and AO's additional objective is evaluated through playable outcomes.

Motivating development comparison and community experience

On November 4, 2026, Ryah reported personal ASCII-versus-AO testing using the same approximately 20k-token prompt and the same model, with 400 messages in each condition: approximately **60% consistency for ASCII** and **95% for Unicode/AO**, a reported difference of about **35 percentage points** [14]. Ryah described ASCII failures as random and unpredictable, and AO failures as more predictable. This is the motivating development A/B observation—an informal natural experiment in the method's development, not a randomized trial. The available account establishes the reported shared prompt/model and run lengths; it does not independently establish an exact glyph-only swap, identical provider settings, randomization, or a fixed scoring rubric. Logs were not reviewed. The approximate rates should not be converted into exact pass/fail counts, and messages within a conversation are correlated rather than independent trials.

Ryah also reports consistent Vexa use personally and within a Discord community of **500+ members**, across models Ryah describes as state of the art [14]. Community size describes the deployment setting, not 500 independently scored participants or controlled replications. This experience is practical behavioral support motivating the theory and its evaluation. It does not by itself isolate which AO component caused the gain or establish the proposed internal attention mechanism.

The development comparison leaves author scoring, run order, divergent histories, sampling, backend changes, and unknown provider quantization as possible influences. Ryah's suggestion that residual failures reflect provider variability remains a candidate explanation. A new evaluation should record these conditions and operationalize consistency before scoring. The available 400-message account does not supply per-turn context lengths, so it cannot locate the onset of an advantage at 100k tokens.

---

9. Evaluation plan

Primary outcome: adherence consistency

Here, **consistency means repeatable adherence to applicable rules and output contracts**. A response may consistently follow a flawed instruction, or contain attractive prose while missing required checks. Score adherence, factual/state correctness against fixtures, and prose quality separately. A claimed route-following explanation is not an execution trace. The primary route evidence is the expected observable consequence of each applicable branch or call.

Resolve contradictory rules in the evaluation fixture first, freeze the expected branch and state changes, and publish that fixture's differences from the source snapshot. Use the same model/version, provider, host, sampler settings, character data, rule set, starting state, and user inputs across conditions. Record unavailable backend details as unknown.

Main experiment: the context-size ladder

Compare the full AO notation with a **one-to-one ASCII-marker substitution** at approximately **30k, 60k, 100k, and 150k assembled input tokens**, within the tested model's supported context limit. Keep the same rules, names, wording, route, and scene checkpoints; freeze the substitution mapping. Change history length using paired, coherent RP history prefixes, not by dropping rules or moving the tested constraint. Count each assembled request with the target tokenizer where available and report actual sizes, output budgets, host truncation, and summarization.

RUNG	PURPOSE	PRIMARY COMPARISON
30k	Lower-context baseline for a 20k+ instruction body	Establish whether either notation already changes adherence
60k	Intermediate history load	Estimate the emerging AO-minus-ASCII gap
100k	Intended long-context stress setting	Test whether AO better preserves applicable checks
150k	Further stress where supported	Test whether the gap persists, widens, or collapses

A feasible **pilot** is four paired scene/history fixtures per rung: **4 rungs × 4 fixtures × 2 notations = 32 generations** for one model/configuration. Randomize or counterbalance condition order. This is a screening design, not a power calculation; report fixture-level results and uncertainty, then add independent fixtures or repeats where needed. Fixed-history checkpoints isolate the notation comparison. Separate rolling continuations test sustained use, but their histories diverge after generation and need conversation-level analysis.

Define the primary contrast before running: **AO adherence minus ASCII adherence at each rung**, and the change in that gap from 30k to 100k/150k. An advantage that widens with context supports the proposed context-dependent traversal benefit. An advantage already present and constant at every rung supports a notation benefit without the predicted context interaction. A gap that vanishes or reverses weakens that claim for the tested setup. Ceiling or floor scores warrant more discriminating fixtures. No single ladder proves a unique internal mechanism; call-removal and routing ablations distinguish its components.

Observable event scoring

Specify expected events and applicability in a machine-readable fixture before generation. A scorer can check structure and deterministic deltas directly; blinded human review handles semantic judgments. These definitions are proposed for the new experiment and do not retroactively redefine Ryah's reported 60%/95% rates.

EVENT FAMILY	EXAMPLE CHECK	MEASUREMENT BOUNDARY
Header and serialization	Header present; all 16 required state sections in order; one complete terminal envelope	Structural check against the frozen template, including explicit empty values
Conditional dispatch	A predefined solo/ensemble or message-trigger case produces its expected scoped effects	Output consequences are evidence of the branch, not direct observation of hidden execution
Calls and local checks	A fixture with a missing prior header restores it; a dropped dormant NPC is restored to state	Check the required repair, rather than a claim that `stateLint` ran
Loop coverage	Every eligible NPC/target receives the required record or deterministic update; ineligible targets do not	Denominator comes from the fixture's eligible set
Null handling	A no-effect case emits the specified empty/null result without inventing an event	Distinguish omission, unsupported action, and valid no-result
Knowledge boundaries	An NPC acts only on information with an established vector in the scene	A cited vector can be fabricated; assess source validity, with semantic review where needed
State retention	Established identifiers and nonchanging values persist; expected deltas occur exactly once	Score against fixture facts, not prose plausibility
Long-response adherence	Applicable constraints hold in early, middle, and late output segments	Report segment scores alongside whole-response scores

Report passed applicable checks divided by all applicable checks, plus an all-required-checks pass rate per response. Publish omissions, wrong-target effects, unsupported knowledge, invalid state, and duplicated/missed updates separately. Estimate variation across independent fixtures or conversations; do not treat all correlated messages or fields as independent samples. Optional brief stage receipts can aid diagnosis in a separate instrumented condition, but change the prompt and do not certify private reasoning. Do not require disclosure of hidden chain of thought.

Component and position tests

PREDICTION	MATCHED INTERVENTION	SUPPORTING OR NARROWING RESULT
Operation legibility adds value	AO versus ASCII, with identical route and definitions	Higher AO adherence supports the chosen notation; parity narrows the glyph-specific claim

PREDICTION	MATCHED INTERVENTION	SUPPORTING OR NARROWING RESULT
Explicit routing adds value	Preserve rule bodies but remove page/call sequencing	Lower adherence without routing supports the traversal component even if ASCII matches Unicode
Local re-entry helps middle sections	Place the same called body near the beginning, middle, or end; cross this with call presence	A smaller middle-position penalty with calls supports local re-entry; unchanged penalties weaken it
Explicit iteration/nulls reduce ambiguity	Remove only loop scope or explicit no-result instructions	More missed targets or invented effects supports those operations
Semantic fit preserves intended Ludus	Compare word labels, word-plus-emoji, emoji-only, and arbitrary-symbol state after the same definition	Fewer unsupported romantic cues with retained playfulness supports the semantic-loading component

The Ludus outcome is adherence to matrix-defined playfulness: a deliberate shift away from an automatic romantic association is success. Hold the numeric matrix, persona, relationship, scene, and model settings fixed. Score friendship/family scenes and Eros-supported romantic scenes separately. Failure means no improvement over matched controls, more unsupported romance, or loss of intended playfulness or context-appropriate expression.

After the primary ladder, compare expanded prose, reconstructed compact base TRP, and AO under equivalent semantics. Run token-budget-matched diagnostics separately from the unchanged-wording notation swap to distinguish compression and context displacement. Record input/output tokens, latency, and cost; assess maintenance time and reference errors in a separate authoring study. Prose quality remains a blinded secondary outcome.

The token comparisons in section 8 are complete. The ladder, event scorer, and ablations above are a proposed behavioral protocol, not completed experiments. A null marker result would not automatically reduce AO to a maintenance tool: explicit routing or local re-entry may still contribute. Those possibilities require their respective comparisons.

---

10. Engineering contribution

TeleWire TRP-AO contributes an implemented **traversal syntax for dense prompts**: route specification, legible operation roles, local re-entry, and a persistent serialization contract. The page/function architecture makes the author's intended processing route explicit. Targeted semantic loading extends that control to how recurring state is interpreted, as in Ludus/ 🎲.

The named path-of-least-resistance hypothesis explains why the design is attention-optimized: the author attempts to make the relevant rule and next operation easier to select at each stage, sustaining adherence during long-form RP as context grows. The implementation establishes this design; the development A/B supplies motivating behavioral evidence; the token fixtures price its representation. The context-size ladder tests the predicted advantage over ASCII, and the ablations test which parts of the traversal account explain it.

The contribution is therefore both an authoring method and a specific behavioral proposition. Its strongest predicted return is more dependable execution of a demanding specification at long context, measured through visible rule and state outcomes.

References and reproducibility

[1] Implementation snapshot. Ryah, *[Lumi] Vexa — Alpha v5*, inspected November 3, 2026. Source artifact: `New folder/[Lumi] Vexa - Alpha v5.json`. SHA-256: `a26276225f8beccd06bfc0cddde916d485541c2cd0607614c13a0547fa2dbd88`.

[2] Dialect comparison. *Observed FrankenSIM / Vexa dialect*, project reference dated November 3, 2026: `.agents/skills/frankensim-frankenwire-syntax/references/vexa-v5-dialect.md`. Records syntax evidence and unresolved issues by block name.

[3] Design background. *The FrankenWIRE / TeleWire TRP Guide*, user-supplied at the FrankenSIM GitHub Wiki, retrieved November 1, 2026. The archived UTF-8 document has SHA-256 `e2c2bbd1b1dff35e8a78c928de32de5435309fbf97d77edcd809f40b087b7ef2`. TRP-AO adapts its compactness-oriented grammar; this revision distinguishes reproduced token counts, Ryah's development observations, and proposed behavioral tests.

[4] Author clarification and notation. On November 3, 2026, Ryah named the method TeleWire TRP-AO, explained its compactness/salience tradeoff, and reported the token ranges above. The supplied NOTATION.md, document revision `c223e8c7a6539e57a063fde135412d0dca40bfe3`, provides mathematical/operator meanings. Snapshot SHA-256: `fb5bc1ab50309e216292f393aac0d424df4d0950c3f720b84059bf5a43450d7a`. It predates portions of the current boot protocol and Antithesis library; current preset definitions take precedence. For example, its two definitions of the all-axis set `Ψ` differ on inclusion of self-axis `ϕ`; that glossary inconsistency is not a new rule for the preset.

[5] Reproduced token examples. `token-evidence.py` and `token-evidence.json`, distributed alongside this paper. Run `python token-evidence.py` with `tiktoken` installed; the recorded version is 0.14.0. Cindy authored the English expansions for this comparison; they are separate from Ryah's historical development examples. No model generation is performed.

[6] Attention formulation. Vaswani et al., *Attention Is All You Need*, section 3.2. Cited to explain attention-weight computation, not as evidence validating TRP-AO.

[7] Tokenization background. Hugging Face, *Tokenization algorithms*. Explains vocabulary-dependent subword and byte-level segmentation; this is why character savings and token savings must be distinguished.

[8] Love-style terminology. John Alan Lee, *A Typology of Styles of Loving*, *Personality and Social Psychology Bulletin*, 3(2), 173–182 (1977); the article also cites Lee's *Colours of Love* (1973). This is related terminology, not a claim that Lee's six styles define the project's twelve independent axes or its intended Ludus combinations.

[9] Targeted semantic loading development account. Ryah's explanation supplied November 3, 2026: the observed Ludus-to-flirting drift, the intended contextual combinations, the definition/emoji intervention, and active development use. Source inspection confirms the one-occurrence legend and emoji-only state instruction in implementation snapshot [1]. Cindy inspected the implementation; the behavioral account is supplied by Ryah.

[10] Reconstructed compact-base comparison. Cindy, November 3, 2026. Report, frozen fixtures, results, and counting script. Seventeen primary pairs frozen before counting, with separate ASCII-marker and state-template diagnostics; `tiktoken` 0.14.0, `cl100k_base` and `o200k_base`. Fixture SHA-256: `a78b7e6ba11f76b616ab56c96e2ef4e48970b1b253e65b8fd2af7acf210bc700`. The base side is Cindy's reconstruction, not a historical base-TRP corpus. Additional Markdown prose fixtures provide the ordinary-instruction baseline; SHA-256: `7561444c00cec52af426c46e81c454068e1fe00292a0ef0e9946cf82993d806d`. They were authored and frozen separately after the original comparison.

Hashes identify the inspected sources without requiring disclosure of the private preset. This paper is a standalone design account; reproducing runtime results would require the implementation, compatible host configuration, and an executed evaluation protocol.

[11] Long-context retrieval. Liu et al., *Lost in the Middle: How Language Models Use Long Contexts*, TACL (2023). Position-sensitive performance in multi-document question answering and key-value retrieval motivates testing rule location in large presets.

[12] Positional attention and context use. Hsieh et al., *Found in the Middle: Calibrating Positional Attention Bias Improves Long Context Utilization*, Findings of ACL (2024). Links positional attention bias to context use and evaluates an attention-calibration intervention; supplies mechanism background for the proposed prompt-level approach.

[13] Re-reading and reasoning. Xu et al., *Re-Reading Improves Reasoning in Large Language Models*, EMNLP (2024). Evaluates repeated question input and compatibility with thought-eliciting prompts. Motivates investigating stage-local rule recall; AO's symbolic references are a distinct proposed application.

[14] Long-context rationale and behavioral development account. Ryah, November 4, 2026: path-of-least-resistance hypothesis; 20k+ instruction bodies and 100k+ narrative RP context target; same-model, same-prompt ASCII/AO comparison of 400 messages per condition with approximately 60% versus 95% reported consistency; qualitative failure-pattern observations; and experience in a Discord community of 500+ members across multiple models. Supplied directly for this revision; logs and scoring data were not independently reviewed.

[15] Traversal revision source check. Cindy re-inspected the unchanged snapshot [1] on November 4, 2026, verifying boot protocol definitions, page/call/return notation, loop/null usage, tail check calls, anchor directives, the state final-check requirement, and the enabled `chat_history` host marker. Ryah supplied the traversal framing and suggested context ladder in the accompanying revision discussion. Static inspection verifies requested operations, not execution or memory allocation.

TeleWire TRP-AO · November 4, 2026
Method and implementation: Ryah · Drafting and token analysis: Cindy
Attention-oriented design theory for large-context presets.