Availability Before Residency: A Free-Theorem Reading of Agent Skill Delivery

Flyxion
Independent Researcher

August 2026

Abstract

Yin et al. propose `@skills`, a protocol for delivering procedural knowledge to AI agents that separates three functions long bundled together under the single operation of "installing" a skill: obtaining its content, keeping a copy of it, and letting it fire without being asked. Their central empirical claim is that only the third of these genuinely requires permanent occupancy of the model's prompt, so the other two can be delivered on demand at zero standing cost. This paper argues that free-theorem reasoning, drawn from Reynolds' relational parametricity and Wadler's popularization of it, supplies a rigorous diagnostic vocabulary for exactly the architectural separation `@skills` discovered empirically, without claiming the protocol is literally an instance of Wadler's theorem. Reading and saving a skill behave, under this vocabulary, like `head` and `fst`: their specifications never refer to where content is stored, so they respect any change of representation without further hypothesis. Auto-triggering, within the prompt-mediated architecture the paper studies, behaves like `sort`: its specification refers to residency directly, so commutation with a representation change requires a hypothesis, that the change preserves residency, which is exactly what the protocol's one-line `.autotrigger` entry purchases. The central claim the analogy is built to support is broader than either function: a bundled interface can force representation-independent operations to pay the cost of the most representation-dependent operation it contains. The paper closes by generalizing that claim beyond skill delivery to any system in which several logically independent operations, some free and some hypothesis-bearing, arrive through a single interface priced at the hypothesis-bearing operation's rate.

1 Introduction

Yin et al. open from a specific, measurable pathology in contemporary agent tooling [1]. Agent skills, procedural knowledge packaged as a directory containing a `SKILL.md` file, are published at scale, 56,804 of them across 1,133 repositories by their July 2026 crawl, yet the dominant way to deliver one to an agent is to install it, after which its description occupies the system prompt permanently, competing for what the authors argue, on independent evidence from the positional-attention literature, is fewer than a hundred reliable slots per agent [4, 5]. The long tail of published skills therefore has no practical path to use, and a team's own working procedures compete for the same scarce room as every public skill anyone has ever published.

Their diagnosis is that installation bundles three logically separate functions: obtaining a skill's content, keeping a copy of it for later, and letting it activate without the user or agent explicitly asking for it. Only the third, they argue, actually needs the resident prompt. The other two can be delivered at the point of use, a path addressed and read exactly when needed, at zero standing

token cost. Their protocol, `@skills`, offers three tiers accordingly: a bare reference that fetches and injects content with nothing left resident afterward; a saved copy, vendored into a project's git-tracked tree, still costing nothing in the prompt because the project's own tooling indexes it outside the model's context; and an installed line, the only mechanism that costs anything, buying exactly one property, firing without being asked.

This paper's claim is that the separation Yin et al. arrive at empirically, by measuring token costs and trigger reliability in a real deployed system, is illuminated by a general and considerably older mathematical fact. Reynolds' theory of relational parametricity, and Wadler's popularization of it as "theorems for free," answers a structurally similar question for polymorphic functions: given an arbitrary transformation of the underlying data a function operates on, when does the function's behavior transport across that transformation automatically, as a consequence of its type alone, and when does it require an explicit hypothesis constraining the transformation [2, 3]? The answer, worked out function by function on a nine-operation catalogue running from `head` through `fold`, is that genuinely parametric operations yield such consequences without any operation-specific preservation assumption, while operations that explicitly consume additional structure, an ordering, an algebra, a predicate, carry a corresponding preservation obligation naming precisely that structure.

Skill delivery is not a polymorphic list function, but the question the `@skills` protocol answers is the same question in a different domain: given a change of representation, here where a skill's content physically lives, on disk, behind a cache, resident in a prompt, does the operation that uses that skill still behave the same way, once its semantic content is held fixed? For reading and saving, physical storage location may vary without affecting the result once content identity is preserved, and the protocol's zero-standing-cost tiers exploit precisely this freedom. For prompt-mediated auto-triggering, no comparable freedom with respect to residency exists, and for a reason exactly parallel to why `sort` needs its ordering hypothesis: the operation's own specification, firing without being asked, is stated in terms of the representation, occupying space the model's attention scans without a query, so no residency-independent formulation of that particular triggering mechanism can exist without replacing the mechanism itself. The protocol's insight that only one of three functions needs residency is, read this way, a correctly identified free-theorem boundary, arrived at from measurement rather than from type theory, but landing in exactly the place the older theory would have predicted it must.

2 Free Theorems: What Commutes, and Why

The relevant machinery is old and can be stated compactly. Given a transformation of an underlying type, $a : A \to A'$, and a polymorphic operation defined uniformly over all instantiations of that type, parametricity supplies a theorem, for free, stating exactly how the operation's behavior relates across the transformation. Writing a^* for the elementwise lifting of a to lists, the simplest cases are unconditional. For `head` and `tail`,

$$a \circ \mathrm{head}_A = \mathrm{head}_{A'} \circ a^*, \qquad a^* \circ \mathrm{tail}_A = \mathrm{tail}_{A'} \circ a^*,$$

which says it does not matter whether one extracts structure first and transforms afterward, or transforms first and then extracts; the two paths agree, for every a, with no further assumption. The same holds for concatenation and flattening, and for `fst`, where the theorem

$$a \circ \mathrm{fst}_{A,B} = \mathrm{fst}_{A',B'} \circ (a \times b)$$

carries a genuine information-flow consequence: the transformation b applied to the discarded component cannot affect the surviving output, because the polymorphic type of `fst` gives it no means of inspecting B at all. These are theorems that come from the type, not from anything an implementer proved by hand; parametricity supplies them automatically for any function with the corresponding polymorphic signature.

The pattern changes with `filter`, `sort`, and `fold`. `filter` still commutes, but only once the predicate itself is transported correctly, $a^* \circ \text{filter}_A(p' \circ a) = \text{filter}_{A'}(p') \circ a^*$; the theorem is unconditional in a only after the predicate has been pulled back along it. `sort` is stricter still: the naturality equation

$$a^* \circ \text{sort}_A(<) = \text{sort}_{A'}(<') \circ a^*$$

holds only under the hypothesis that a preserves the order, $\forall x, y \in A,\ (x < y) = (ax <' ay)$. Without that hypothesis there is no theorem at all; an arbitrary a can scramble the order and `sort` will simply disagree across the transformation. `fold` requires the strongest hypothesis of the nine, that the transformation on the accumulator respects the combining operation and the base case, $b(x@y) = (ax)@'(by)$ and $bu = u'$, an algebra-homomorphism condition rather than a naturality square.

Proposition 1 (Parametricity principle)**.** *For a genuinely parametric operation F, transformations related by the relational interpretation of its polymorphic type induce corresponding related outputs, unconditionally. Additional structure explicitly supplied to the operation, a predicate, an ordering, an algebra, introduces a corresponding preservation condition that the transformation must separately satisfy before the same conclusion follows.*

The examples above exhibit a recurring distinction along exactly these lines: genuinely parametric operations yield relational consequences without operation-specific preservation assumptions, whereas operations supplied with additional structure, an ordering, a predicate, an algebra, carry corresponding conditions on transformations of that structure. `head` names only positional identity. `sort` names an order. Which pattern an operation falls into is fixed the moment its specification is written down, before any proof is attempted.

3 Skill Reference and Save as Free Theorems

The `@skills` protocol's first two tiers can be stated in exactly this vocabulary. Let a skill's representation be a pair (loc, mat), its location, local disk, a git remote, a cache entry, and its materialization, the byte-for-byte content at that location. Let RESOLVE be the operation that, given a path identifying a skill, returns its content for injection at the point of use, as specified by the protocol's resolution algorithm: consult the local project tree first, and failing that, resolve through a validating cache against the remote [1]. Define two representations observationally equivalent when they carry the same content,

$$r \sim r' \iff \text{content}(r) = \text{content}(r').$$

Proposition 2. RESOLVE *is invariant over the observational-equivalence classes induced by content: $r \sim r' \implies \text{RESOLVE}(r) = \text{RESOLVE}(r')$. Once representations are restricted to a common content-equivalence class, no additional condition on physical storage location is required for this to hold.*

Equivalently, RESOLVE factors through the quotient $R/\!\sim$: at the level of observable behavior, representations differing only in storage location, local disk, git remote, cache entry, denote the same semantic object. This is the formal content behind the protocol's claim that "a path addresses

any skill…and reading it is using it, so nothing installs and nothing becomes resident" [1]. The proposition is not a discovery so much as a recognition: RESOLVE's specification refers only to identity (which path names which content) and never to residency (where in the agent's memory the content happens to sit while being used). The protocol's validating cache, git-based transport, and local-first resolution rule are all engineering realizations of a representation that was already, by the specification's own silence on the matter, free to vary.

The save operation is the closer analogue of `fst`, though the analogy needs to be stated carefully: an arbitrary function of type $A \times B \to A$ could in principle inspect B before discarding it, so noninterference is not automatic merely because the state resembles a product. What licenses the comparison is that the protocol deliberately specifies resolution as a projection with respect to provenance: a saved skill carries two components, the content itself, vendored into the project's tree, and a provenance stamp recording where it came from and at what revision, and the protocol states directly that the stamp is written once and never consulted during resolution [1]. Writing $\text{SAVE}(id) = (content, provenance)$, once resolution is architecturally constrained to be a projection onto the content component, the noninterference of provenance follows in the same way `fst`'s noninterference follows from its own definition as a projection, and not from the mere fact that a product type is involved. The provenance stamp remains meaningful to a human auditing history while being structurally prevented, by that design choice, from affecting what a later reference to the same path returns.

4 Auto-Triggering as the Sort Case

Auto-triggering is where the analogy stops being a restatement and starts doing work, because within the specific architecture the paper studies, prompt-mediated triggering, it is the one operation whose specification cannot be written without naming the representation. Define $\text{TRIGGER}(s)$ to hold for a skill s if the agent activates s in response to a request without s being named explicitly, and define residency, $\text{RES}(s)$, to hold if s's frontmatter occupies the prompt at session start, before any request has been made. The claim under examination is narrower than a general fact about transformer architecture, and the source paper is itself careful to state its bound on reliable trigger slots as an argument from evidence and literature rather than a direct measurement [1]. Within the mechanism the protocol actually implements, installed descriptions resident in the prompt, matched probabilistically against a request, with bodies loading only on trigger, the paper's own evidence on distance decay, standing token cost, and dilution among competing descriptions [4, 5, 6] establishes that $\text{TRIGGER}(s)$ is conditioned on $\text{RES}(s)$: a skill whose description is not resident at the moment the model reads a request has no path, within this mechanism, to matching that request unprompted. Residency is not logically inherent in the abstract notion of automatic triggering; an architecture built instead around an external classifier, a retrieval index, or a routing daemon could in principle decouple the two. The claim defended here is the narrower and better-supported one: within prompt-mediated implicit matching specifically, the mechanism the protocol was built to serve, triggering requires resident trigger information.

Proposition 3. *Within the prompt-mediated triggering architecture,* TRIGGER *does not commute with an arbitrary transformation a of a skill's storage representation. It commutes only under the hypothesis that a preserves residency,* $\text{RES}(s) \Rightarrow \text{RES}(a(s))$*, which is not a property of a in general but a specific structural commitment that must be separately purchased.*

This is formally the same shape as the `sort` theorem's order-preservation hypothesis. Sorting is not defined over bare elements; it is defined over an ordered type, and an arbitrary transformation

that does not respect the order breaks the theorem, not because sorting was implemented carelessly but because the operation's specification refers to structure the transformation was never asked to preserve. Auto-triggering, within this architecture, is not defined over bare content; it is defined over a resident prompt, occupying attention the model allocates without being asked, and an arbitrary storage transformation that does not preserve residency breaks the theorem for the identical reason. The protocol's `.autotrigger` file is, in this reading, the explicit hypothesis `sort` requires, written out as an engineering artifact: one line per skill for which the residency-preservation hypothesis has been deliberately purchased, and the token cost the paper measures, fifty to a hundred tokens per resident frontmatter, is the price of discharging that hypothesis rather than an arbitrary tax.

The comparison sharpens a claim the original paper makes more informally, that "installing bundles three separable functions...and only the last needs the prompt" [1]. The free-theorem reading explains why the separation holds rather than merely observing that it happens to hold in their measured system. Reading and saving have specifications silent on residency, so representation changes that preserve their relevant observational content require no additional residency-preservation condition, and no protocol design choice could make residency necessary for them without adding a clause to their specification that was never there. Auto-triggering, within the prompt-mediated architecture under consideration, has a specification that mentions residency by definition, firing without being asked is stated in terms of what the model has already been shown, so within that architecture, eliminating residency would require replacing rather than merely reorganizing the triggering mechanism. What the decomposition establishes is a lower bound on representational commitment for each operation, not a uniqueness claim about the resulting architecture: reference and save require addressability and persistence but no prompt residency, while prompt-mediated implicit triggering additionally requires resident trigger information, and `@skills` realizes that particular decomposition unusually cleanly, exactly as its own description of the three tiers states.

5 Composition: Selection Determinism, Not Risk Elimination

One further parallel is worth making precise, narrowed to what the source paper actually supports rather than what the fold analogy might suggest at first pass. The paper treats deterministic composition of several skills referenced in one message as a practical advantage over the "trigger lottery" of several installed descriptions each needing an independent probabilistic match [1]. `fold`'s free theorem requires that the transformation on the accumulator respect the combining operation, $b(x@y) = (ax)@'(by)$; once that hypothesis is granted, folding a sequence commutes with the transformation uniformly across composition, regardless of length or grouping.

The claim this licenses about skill composition is narrower than a statement about eliminated risk overall. Resolving several references in one message can still fail for reasons the parametricity argument says nothing about: network unavailability, malformed content, conflicting instructions between skills, or security concerns in fetched text. What the fold analogy helps characterize is selection determinism specifically: because each reference names its own path explicitly, which skills co-fire in a given message is settled by the message itself, the way a fold's decomposition is settled by the sequence's structure once the algebra hypothesis holds, rather than by n independent probabilistic matches each of which might fail regardless of what the others do. The determinism itself follows from explicit naming, not from the fold theorem; the analogy's contribution is a vocabulary for why that determinism composes cleanly across an arbitrary number of references rather than degrading, the same way a hypothesis once discharged for `fold` propagates uniformly across composition instead of being re-earned at each step. Installed, auto-triggered skills co-fire only if each of n separate matches against a resident description succeeds; explicit references co-fire

because each was named, a distinction in which skills get selected, not a guarantee about what happens once they do.

A natural extension is worth developing rather than merely flagging, since it tests whether the framework can distinguish architectures rather than only describe the one at hand.

6 Retrieval as Pullback

Section 4 established that within prompt-mediated architectures, TRIGGER behaves like `sort`: its specification requires residency, an order-like structural condition that a storage transformation must separately preserve. Whether that dependence is inherent to automatic activation as such, or an artifact of the specific mechanism the source paper studies, is worth testing against a different triggering architecture rather than left as an open question.

Consider a retrieval-mediated design: a skill's description does not sit permanently in the prompt; instead an external index, embeddings, a vector store, a search structure, evaluates an incoming request against candidate skills and injects only those a similarity or matching criterion selects. Let p_{query} be the predicate determining whether a skill is relevant to a given request, evaluated over the index rather than over resident prompt content. Recall that `filter` commutes across a representation change a once the predicate is transported correctly along it, $a^* \circ \text{filter}_A(p' \circ a) = \text{filter}_{A'}(p') \circ a^*$; the theorem does not ask a to preserve any structure as rigid as an ordering, only that the predicate deciding membership is re-indexed consistently with the transformation.

Here p_{query} denotes the extensional selection predicate induced by the complete retrieval procedure, abstracting over similarity scores, thresholds, top-k selection, and index implementation. Real embedding retrieval is not literally Boolean, and does not transport a predicate exactly under an arbitrary representation change; approximate nearest-neighbor search, model updates, and tie-breaking all introduce their own sources of drift. The proposition below states the ideal extensional condition under which selection is invariant; actual retrieval systems approximate it to varying degrees, and the gap between the two is itself diagnostic of how far a given implementation is from behaving as a clean `filter`.

Proposition 4. *Under a retrieval-mediated architecture,* TRIGGER_{RAG} *does not require residency* $\text{RES}(s)$*. Commutation across a change of storage representation holds provided the extensional retrieval predicate is transported along the transformation,* $p'_{query} \circ a = p_{query}$*; where that transport fails to hold, so does commutation, and the failure is observable as a skill silently entering or leaving the trigger set incorrectly.*

The dependence has not been eliminated; it has migrated. Prompt-mediated triggering requires

$$\text{storage transformation} + \text{residency preservation} \implies \text{trigger preservation},$$

while retrieval-mediated triggering requires

$$\text{content transformation} + \text{predicate transport} \implies \text{selection preservation}.$$

The expensive commitment moves from standing token occupancy to index maintenance: if a skill's content changes without a corresponding update to its embedding or index entry, the predicate is no longer correctly transported, and the failure is exactly what practitioners call index staleness, a skill that should trigger silently does not, or one that should no longer trigger continues to. This is not a cache-invalidation detail incidental to retrieval; it is the retrieval architecture's own version of the preservation obligation that `.autotrigger` makes explicit for the prompt-mediated case, now attached to the index rather than to the prompt. A retrieval-based system trades the standing token

tax of Section 4 for a continuous synchronization obligation between content and index, decoupling standing prompt cost from library size, since growth of the retrieval corpus need not produce corresponding growth of resident prompt context, but reintroducing a probabilistic boundary at the point of retrieval rather than restoring the unconditional selection determinism of Section 5. `@skills`'s choice of direct path reference over built-in retrieval can be read, in this vocabulary, as choosing the zero-hypothesis territory of `head` and `fst` over either the residency hypothesis of `sort` or the transport hypothesis of `filter`, at the cost of leaving discovery, finding a skill one does not already know the path to, outside what the reference tier itself can offer, which is exactly the gap the source paper's companion hub is built to fill [1].

The general shape underneath both cases is what makes the framework more than descriptive vocabulary for one system: an operation's standing cost tracks whichever structure its specification requires a representation change to preserve, and different architectures for the same function, prompt residency versus index synchronization, are different choices of what that structure is, not different answers to whether some structure must be preserved at all.

7 The General Pattern

Stripped of its origin in either list functions or skill directories, the underlying claim is a fact about bundled operations generally, but stating it precisely requires one further distinction that Section 3 already implicitly relies on. Every operation with any semantic content has *some* invariant it cannot survive violating: RESOLVE plainly does not survive arbitrary corruption of a skill's content, since content identity is exactly what it is specified to preserve. Write $P_{\text{sem}}(O)$ for this minimum observational invariant, the content-equivalence class $\sim$ of Proposition 2 in the case of RESOLVE, an invariant every architecture for that operation must respect on pain of no longer being that operation at all. The question this paper has actually been asking is narrower and more useful: holding $P_{\text{sem}}(O)$ fixed, what *additional*, representation-specific obligation, write $P_{\text{rep}}(O)$, does an operation impose on a change of physical representation beyond that minimum? This is exactly the question the original catalogue of free theorems answers function by function: `head` demands no ordering; `sort` does; `filter` demands correspondence of predicates; `fold` demands preservation of an algebra. The four cases developed above now read cleanly in these terms,

$$\begin{aligned} P_{\text{rep}}(\text{RESOLVE}) = P_{\text{rep}}(\text{SAVE}) &= \varnothing, \\ P_{\text{rep}}(\text{TRIGGER}_{\text{prompt}}) &= \{\text{residency}\}, \\ P_{\text{rep}}(\text{TRIGGER}_{\text{RAG}}) &= \{\text{index consistency}\}, \end{aligned}$$

with P_{sem} nonempty and shared across all four, content identity in every case, joined by persistence for SAVE. The claim that reading and saving "incur no obligation at all" was always a claim about P_{rep}, never about P_{sem}; nothing in this paper suggests RESOLVE survives its skill being replaced with unrelated content, only that it survives that content moving between disk, cache, and remote.

This distinction sharpens commitment-honesty into something considerably stronger than a restatement of the residency case. A commitment-honest system is one in which cost tracks $P_{\text{rep}}(O)$, the contingent representational commitment, not $P_{\text{sem}}(O)$, which every implementation must pay regardless of architecture, and not P_{bundle}, the union of representational commitments across whatever else happens to be packaged alongside O,

$$C(O) \propto P_{\text{rep}}(O), \qquad \text{not} \qquad C(O) \propto P_{\text{rep}}(\text{bundle}).$$

This closes the obvious objection that nothing is ever truly representation-independent, since something always has to be preserved: the claim was never that P_{sem} can be zero, only that P_{rep}

can be, and that a well-designed interface charges for the latter rather than folding the former's unavoidable cost into a justification for taxing every bundled operation at the rate of whichever one has the largest P_{rep}.

The four representational commitments above are also not simply ordered from cheap to expensive; they form a partial structure worth noting without a full digression into it. SAVE's commitment strictly contains RESOLVE's, $P_{\text{rep}}(\text{RESOLVE}) \subsetneq P_{\text{rep}}(\text{SAVE})$, since persistence is genuinely additional structure beyond reference. But residency and index consistency are not comparable in this way: $\{\text{residency}\} \not\subset \{\text{index consistency}\}$ and $\{\text{index consistency}\} \not\subset \{\text{residency}\}$, since prompt-mediated and retrieval-mediated triggering purchase the same semantic function, unprompted activation, through structurally different, mutually irreducible commitments, rather than one strictly subsuming the other. Architectures are therefore better compared by inclusion between their representational commitments than by a single scalar notion of cost.

When several operations $O_1, \ldots, O_n$ are packaged behind one interface, an interface such as "install," the interface's bundle obligation tends toward the union of representational commitments,

$$P_{\text{rep}}(\text{bundle}) \supseteq \bigcup_{i=1}^{n} P_{\text{rep}}(O_i).$$

Proposition 5 (Bundling pathology)**.**

Bundling pathology occurs when $P_{rep}(O_i) \subsetneq P_{rep}(bundle)$ but O_i is charged for the latter.

A user paying the cost of installation pays for $P_{\text{rep}}(\text{bundle}) \supseteq \{\text{residency}\}$ on every skill, including the overwhelming majority whose own P_{rep}, read or save, is empty. The paper's central empirical finding, that fewer than a hundred slots must serve tens of thousands of skills, is the visible symptom of exactly this leakage: an operation with $P_{\text{rep}}(O) = \varnothing$ and one with $P_{\text{rep}}(O) \neq \varnothing$ were sold at the same price because they arrived through the same door.

Definition 1. *Call a system* commitment-honest *if, for every operation O_i it exposes, the cost a user pays for using O_i is proportional to $P_{rep}(O_i)$ rather than to $P_{rep}(bundle)$, the union of representational commitments across whatever other operations happen to be bundled with it.* Residency-honest*, as used above, is the instance of this definition specific to prompt occupancy; the general principle is*

cost of an operation $\longleftrightarrow$ representational structure that must be preserved for it to remain valid across a representation change, beyond whatever semantic invariant the operation could never survive losing.

The `@skills` protocol is, on this definition, a commitment-honest redesign of a commitment-dishonest predecessor, and the retrieval comparison of Section 6 shows the definition doing real work beyond the single case it was drawn from: prompt residency and index synchronization are two different, incomparable instances of $P_{\text{rep}}(\text{TRIGGER})$ for the same underlying function, and a system is commitment-honest to the extent it charges each operation for the $P_{\text{rep}}(O_i)$ its own specification actually incurs, rather than for whichever $P_{\text{rep}}(\text{bundle})$ happens to be attached to the interface it arrived through. The free-theorem framing gives a general method for locating the next instance of the same pathology elsewhere: identify the bundled interface, write down $P_{\text{rep}}(O_i)$ for each bundled operation honestly, and check which are empty and which are not. Operations with $P_{\text{rep}}(O_i) = \varnothing$ are owed representation-independence beyond their unavoidable semantic invariant, and any further cost they are currently paying is a design defect rather than a physical necessity. Operations with $P_{\text{rep}}(O_i) \neq \varnothing$ are owed an explicit, visible obligation, stated as plainly as `sort`'s

order-preservation condition, `filter`'s predicate transport, or the protocol's own `.autotrigger` line, so that the cost of discharging it is paid exactly once, by exactly the operations that need it. Representation changes are cheap or expensive not in themselves, but according to which operational invariants they are required to preserve, and bundling becomes pathological precisely when the representational commitment of one operation is imposed on operations whose own invariants do not require it.

8 Conclusion

Yin et al. arrive, through measurement of a deployed system, at a three-way separation between operations that need no representational commitment and one that needs a specific and costly one. This paper has argued that free-theorem reasoning, properly stated, supplies a diagnostic vocabulary for that separation rather than proof that the protocol is literally an instance of Wadler's theorem: parametric operations can yield representation-independence consequences without operation-specific preservation assumptions, while operations that explicitly consume additional structure carry corresponding preservation obligations. Reading and saving a skill are, in this sense, the `head` and `fst` of agent tooling, representation-free by the shape of what they are specified to do. Auto-triggering, within the prompt-mediated architecture under study, is its `sort`, carrying a residency-preservation obligation by the same logic; a retrieval-mediated alternative relocates that obligation to something closer to `filter`'s predicate transport, trading a standing token cost for a continuous index-synchronization one rather than eliminating a cost altogether. The general lesson is the sentence this whole comparison was built to support: a bundled interface can force representation-independent operations to pay the cost of the most representation-dependent operation it contains. That claim does not depend on overstating what parametricity proves, generalizes past skill delivery and past any single triggering mechanism to any system with the same bundling defect, and is what makes the analogy to a decade-old sheet of naturality laws useful rather than decorative.

References

[1] L. Yin, Z. Li, Z. Shi, H. Zhang, H. Seong, and Z. Wang. @skills: Attention Is All You Have — An Open Agent Skills Protocol, Built for the Future of Knowledge Sharing. *arXiv preprint arXiv:2608.12610*, 2026.

[2] J. C. Reynolds. Types, Abstraction and Parametric Polymorphism. In *Information Processing 83*, pages 513–523. North-Holland, 1983.

[3] P. Wadler. Theorems for Free! In *Proceedings of the Fourth International Conference on Functional Programming Languages and Computer Architecture (FPCA)*, pages 347–359. ACM, 1989.

[4] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[5] P. Laban, H. Hayashi, Y. Zhou, and J. Neville. LLMs Get Lost in Multi-Turn Conversation. *arXiv preprint arXiv:2505.06120*, 2025.

[6] D. Jaroslawicz, B. Whiting, P. Shah, and K. Maamari. How Many Instructions Can LLMs Follow at Once? *arXiv preprint arXiv:2507.11538*, 2025.