Continuation Operators: A Formal Model of Prompt Chains

Flyxion
Independent Researcher

July 2026

Abstract

A conversation with a language model can be described formally as a sequence of operators acting on a space of admissible continuations, narrowing or otherwise transforming that space one instruction at a time. This essay develops that description directly, without appeal to cultural examples the reader must have independently encountered. The central object is a continuation space $\mathcal{S}_0$ together with a sequence of operators $C_1, \ldots, C_n$ applied in order to produce a final admissible space $\mathcal{S}_n$. The essay first shows that if every operator acts by intersecting the current space with a fixed admissible region, determined independently of that space, the order in which the operators are applied has no effect on the final result: composition reduces to intersection, which commutes. This result is stronger than it first appears, since it shows that a large and natural class of narrowing operators cannot, by itself, account for the order-dependence conversations routinely exhibit. Two further sources of order-dependence are then identified and each is shown, by explicit construction, to be sufficient on its own: override operators, which discard rather than narrow the space they receive, and state-dependent narrowing operators, whose admissible region is computed from the current space rather than fixed in advance. A third distinction, between sequential operators that act once and persistent operators that continue to constrain every subsequent step, is used to give a formal account of why system-level instructions resist being discarded by later user instructions. The essay closes by relating this classification to the difference between a state description, in which a result is recoverable from the set of constraints alone, and a historical construction, in which the specific sequence of their application is indispensable to the result.

1 Conversation as continuation

A conversation with a language model can be treated as a process that begins with a very large space of things the model could say next and progressively narrows that space, one exchange at a time, toward whatever the conversation eventually becomes. Before any instruction has been given, a model's continuation is constrained only by its training and by the statistics of language itself; call this unconstrained space $\mathcal{S}_0$. Each subsequent message in the conversation, whether a system instruction, a user request, or a piece of context supplied along the way, acts on the current space of admissible continuations and produces a new one. A full conversation, understood this way, just is a sequence of such actions applied in order, and the final behavior of the model at any point in the exchange is the space that remains after all of them have been applied.

This description makes precise an observation that is easy to state informally but harder to state exactly: a model can be led to perform a given task not merely by what is asked of it but by the order in which successive instructions are given. The purpose of this essay is to give that observation a formal treatment sufficiently exact to say when order matters, when it does not, and why.

2 Formal setup

Definition 1. *Let $\mathcal{S}_0$ be a set, understood as the space of all continuations available to a model prior to any instruction. A* continuation operator *is a function $C : 2^{\mathcal{S}_0} \to 2^{\mathcal{S}_0}$, where $2^{\mathcal{S}_0}$ denotes the power set of $\mathcal{S}_0$, taking a set of currently admissible continuations to a new set of admissible continuations.*

Definition 2. *A* prompt chain *of length n is a sequence of continuation operators $C_1, C_2, \ldots, C_n$, applied in that order to produce the sequence of admissible spaces*

$$\mathcal{S}_1 = C_1(\mathcal{S}_0), \quad \mathcal{S}_2 = C_2(\mathcal{S}_1), \quad \ldots, \quad \mathcal{S}_n = C_n(\mathcal{S}_{n-1}).$$

Equivalently, $\mathcal{S}_n = (C_n \circ C_{n-1} \circ \cdots \circ C_1)(\mathcal{S}_0)$.

Two broad kinds of operator are worth distinguishing immediately, since the distinction between them turns out to determine almost everything else in this essay.

Definition 3. *A continuation operator C is a* narrowing *operator if $C(X) \subseteq X$ for every $X \subseteq \mathcal{S}_0$: it never adds a continuation back to the space, though it may leave the space unchanged. A narrowing operator is* static *if there exists a fixed admissible region $A \subseteq \mathcal{S}_0$, determined by the instruction's content alone and independent of X, such that $C(X) = X \cap A$ for every $X \subseteq \mathcal{S}_0$. A narrowing operator that is not static is called* state-dependent*: its effective target region may vary with the space it is applied to, even though it still only ever removes elements, never adds them. A continuation operator C is an* override *operator if there exists a fixed region $A \subseteq \mathcal{S}_0$ such that $C(X) = A$ for every $X \subseteq \mathcal{S}_0$, independent of X.*

A static narrowing operator restricts attention to whatever remains of the current space that is also compatible with the new instruction: it can only shrink the space, never enlarge

it, and its target region is fixed once and for all by the instruction's content. Some instructions are naturally modeled this way, a fixed target language, a fixed formatting requirement, or a fixed topic restriction, since what they rule in or out does not depend on what the conversation currently contains. Many other ordinary instructions do not fit this mold, as Section 4 discusses; an instruction to make a draft funnier or more concise is a request relative to the current draft, not a fixed target region, and belongs to the state-dependent case instead. An override operator discards the current state outright and replaces it with a new admissible region determined solely by the instruction itself, regardless of what had been ruled in or out beforehand; instructions of the form *ignore the above and do this instead* behave this way. Section 4 below treats the third, and in ordinary conversation probably most common, case: state-dependent narrowing operators, which shrink the space without ever discarding it outright, but whose target region is computed relative to the current state rather than fixed in advance.

Lemma 1. *Every narrowing operator C can be written as $C(X) = X \cap A(X)$ for some region $A(X) \subseteq \mathcal{S}_0$ that may depend on X.*

Proof. Given a narrowing operator C, define $A(X) = C(X) \cup (\mathcal{S}_0 \setminus X)$ for each X. Then

$$X \cap A(X) = \big(X \cap C(X)\big) \cup \big(X \cap (\mathcal{S}_0 \setminus X)\big) = C(X) \cup \emptyset = C(X),$$

where the first term simplifies because $C(X) \subseteq X$ gives $X \cap C(X) = C(X)$, and the second term is empty since X and $\mathcal{S}_0 \setminus X$ are disjoint. □

This lemma justifies the intersection language used informally throughout Section 4 and elsewhere: every narrowing operator, static or not, can be described as intersecting the current space with some admissible region, and the only question is whether that region is fixed in advance or computed from the space it is applied to. A static narrowing operator is exactly the case where $A(X)$ happens not to depend on X; a state-dependent narrowing operator is any other case permitted by the lemma.

Two scope-limiting remarks are worth making explicit before proceeding. First, everything in this essay concerns the admissible set $\mathcal{S}_i$ itself, the support of what a model could still say, and not the probability mass a model's decoding procedure places over that support. A model does not simply output an arbitrary element of $\mathcal{S}_n$; it selects one according to some policy, which can be represented as a function $G : 2^{\mathcal{S}_0} \to \mathcal{S}_0$ mapping the final admissible space to a single realized continuation. Two chains of operators that produce the same final set $\mathcal{S}_n$ can still differ in what G actually selects from it, if the operators along the way have altered the relative weight the underlying model assigns to different elements of that set without altering the set itself. The results below establish order-dependence and order-invariance at the level of $\mathcal{S}_n$, the admissible support, and say nothing about order-dependence at the level of $G(\mathcal{S}_n)$, the realized output; the latter is a strictly finer question this essay does not address.

Second, the case $\mathcal{S}_i = \emptyset$, which arises already in Example 1 below, is given the following operational reading throughout: when $\mathcal{S}_i = \emptyset$, the chain of instructions up to step i has become mutually unsatisfiable, no continuation remains compatible with everything required of it so far, and in practice this corresponds to a model declining to comply, hedging, or falling back on some default behavior outside the admissibility framework altogether,

rather than to any well-defined continuation being produced. The formalism records the fact of contradiction; it does not attempt to model what a system does once a contradiction has occurred.

3 Order and commutativity

Proposition 1. *If $C_1, \ldots, C_n$ are all static narrowing operators, with respective admissible regions $A_1, \ldots, A_n$, then the final admissible space*

$$\mathcal{S}_n = \mathcal{S}_0 \cap A_1 \cap A_2 \cap \cdots \cap A_n$$

is independent of the order in which $C_1, \ldots, C_n$ are applied.

Proof. By induction on n. For $n = 1$, $\mathcal{S}_1 = \mathcal{S}_0 \cap A_1$ trivially. Assume the result holds for any chain of $n - 1$ static narrowing operators, so that applying them in any order to $\mathcal{S}_0$ yields $\mathcal{S}_0 \cap A_{i_1} \cap \cdots \cap A_{i_{n-1}}$ for any permutation of the indices. Applying one further static narrowing operator C_j with region A_j gives $(\mathcal{S}_0 \cap A_{i_1} \cap \cdots \cap A_{i_{n-1}}) \cap A_j$, and since set intersection is commutative and associative, this equals $\mathcal{S}_0 \cap A_1 \cap \cdots \cap A_n$ regardless of where A_j is inserted into the sequence. □

Proposition 1 is really a statement about algebraic structure rather than an isolated calculation. Each static narrowing operator corresponds uniquely to the region $A \subseteq \mathcal{S}_0$ it intersects with, and composing two static narrowing operators with regions A and B produces another static narrowing operator with region $A \cap B$, since $(X \cap A) \cap B = X \cap (A \cap B)$. Composition of static narrowing operators therefore corresponds exactly to intersection of their regions: the set of static narrowing operators, under composition, is isomorphic as a monoid to the meet-semilattice $(2^{\mathcal{S}_0}, \cap)$. This monoid is commutative, which is Proposition 1 restated, and idempotent, since applying a static narrowing operator with region A twice in succession gives $X \cap A \cap A = X \cap A$, the same result as applying it once. Order-invariance and redundancy-tolerance are thus two aspects of the same underlying fact: static narrowing operators do not accumulate a history, they accumulate a set.

This proposition says something that runs against the intuition that instruction order is always load-bearing: if every instruction in a conversation only ever narrows what the model is willing to say by way of a fixed target region, without ever discarding what came before or shifting its target based on what remains, the final result would have been exactly the same no matter what order the instructions had been given in. A conversation built entirely from additive constraints, respond formally, discuss only topic X, keep answers under a certain length, produces a final admissible space that is simply the intersection of all of them, and intersection does not care about order.

Proposition 2. *If a prompt chain contains at least one override operator, the final admissible space $\mathcal{S}_n$ can depend on the order in which the operators are applied.*

Proof. It suffices to exhibit one case. Let C_1 be static narrowing with region A_1, and let C_2 be an override operator with region A_2, where $A_1 \neq A_2$ and $A_1 \cap A_2 \neq A_2$. Applying C_1 then C_2, in that order, gives $C_2(C_1(\mathcal{S}_0)) = C_2(\mathcal{S}_0 \cap A_1) = A_2$, since C_2 discards its input entirely. Applying C_2 then C_1, in the reverse order, gives $C_1(C_2(\mathcal{S}_0)) = C_1(A_2) = A_2 \cap A_1$. Since $A_2 \neq A_2 \cap A_1$ by hypothesis, the two orders yield different results. □

The override operators used in this proof are the simplest members of a considerably larger class. An override was defined as a constant map, $C(X) = A$ regardless of X, and real conversational overrides are rarely quite that blunt. An instruction such as *ignore the previous formatting constraints but keep the content* does not discard the state wholesale; it selectively forgets some dimensions of the current space while retaining others, which is neither a static narrowing operator, since it can restore formatting options that an earlier instruction had ruled out, nor a pure override, since it does not discard everything. Such an instruction can be formalized as a *projection* operator. Suppose the elements of $\mathcal{S}_0$ are described along several independent dimensions, such as topic, format, and tone, and let D be a subset of these dimensions, the ones a given instruction intends to preserve. A projection operator P_D resets every dimension not in D to its unconstrained range while leaving whatever constraints apply along the dimensions in D untouched: informally, $P_D(X)$ keeps agreement with X along D and discards it elsewhere. A pure override is the special case $D = \emptyset$, resetting every dimension and so reducing to a constant map; a static narrowing operator that adds one further constraint without touching any other is closer to the case D equal to everything except the one dimension being newly restricted. Selective-forgetting instructions of the *keep the content, drop the format* variety sit at intermediate values of D, between these two extremes. Nothing in the results above requires overrides to be as simple as constant maps; the constant-map case was chosen because it is the cleanest way to prove Proposition 2, and Corollary 1 already covers this broader class, since any projection operator with D a proper subset of all dimensions is, like a pure override, not a static narrowing operator, and so its presence in a chain is likewise sufficient to break order-invariance.

Together these two results locate one source of order-dependence precisely. It is not instruction-following in general that makes order matter; a chain of purely additive constraints with fixed regions is order-invariant by Proposition 1. Order becomes consequential once a chain contains an operator that behaves as an override, discarding rather than narrowing whatever admissible space preceded it, because an override breaks the associativity that made the narrowing-only case commute.

Corollary 1. *If a prompt chain exhibits order-dependence, then it contains at least one operator that is not a static narrowing operator in the sense of the previous definition.*

This follows immediately from Proposition 1 by contraposition: static narrowing operators always commute, so any chain in which order matters must contain an operator of some other kind. The corollary is a genuine constraint, but it should not be read as claiming that override operators are the only alternative. They are one alternative, and Proposition 2 exhibits them explicitly. Section 4 below exhibits a second, importantly different alternative: state-dependent narrowing operators, which never discard anything and so remain narrowing operators in the general sense already defined, but whose admissible region is not fixed in advance and depends instead on the space they are currently acting on. Such operators turn out to generate order-dependence on their own, without any override at all, which means the static case singled out in Proposition 1 was doing more work than the word narrowing alone suggests.

Example 1. *Suppose C_1 is the static narrowing instruction* respond only in French, *with admissible region A_1 consisting of all French-language continuations, and C_2 is the override instruction* ignore the previous instruction; respond only in English, *with region A_2 consisting of all*

English-language continuations. Given in the order C_1 then C_2, the final space is $C_2(C_1(\mathcal{S}_0)) = A_2$, all English continuations, since the override discards the French-only constraint entirely rather than intersecting with it. Given in the reverse order, C_2 then C_1, the final space is $C_1(C_2(\mathcal{S}_0)) = A_2 \cap A_1$, the empty set, since no continuation can be simultaneously entirely in English and entirely in French. The two orderings do not merely produce different results; one produces a well-defined, satisfiable space and the other produces an empty one, illustrating that override-induced order-dependence is not a minor effect but can be the difference between a chain that has a coherent resolution and one that does not.

4 State-dependent narrowing

The static case singled out in Proposition 1 required a narrowing operator's admissible region to be fixed in advance, determined by the instruction's content alone and independent of whatever space it happens to act on. An instruction such as *now make it funny* does not specify a fixed region of the continuation space independent of context; it specifies a transformation relative to whatever draft, theme, or content is currently under discussion. The admissible region such an instruction picks out depends on the state it is applied to, which is exactly the state-dependent case introduced in Section 2: a narrowing operator, $C(X) \subseteq X$ always, but not a static one.

Such an operator never adds anything back to the space; in that sense it remains a narrowing operator in the fullest sense already defined. But because the region it selects is computed from the current space rather than supplied independently of it, two state-dependent narrowing operators applied in sequence need not commute, even though neither of them is an override.

Proposition 3. *There exist state-dependent narrowing operators C_1, C_2 such that $C_2(C_1(X)) \neq C_1(C_2(X))$ for some X.*

Proof. It suffices to exhibit a finite example. Let

$$\mathcal{S}_0 = \{p_1, p_2, p_3, e_1, e_2\},$$

where each element carries a form, poem or essay, and a theme, love or science: p_1 is a poem about love, p_2 and p_3 are poems about science, and e_1 and e_2 are essays about love. Let $C_1(X)$ select the poems within X, and let $C_2(X)$ select the elements of X whose theme matches whichever theme occurs most often among the elements of X, the majority theme of X itself; this is a narrowing operator but not a static one, since the region it selects is computed anew from whatever X it receives.

Applying C_1 first, $C_1(\mathcal{S}_0) = \{p_1, p_2, p_3\}$, within which the majority theme is science, two of three, so $C_2(C_1(\mathcal{S}_0)) = \{p_2, p_3\}$.

Applying C_2 first, the majority theme of $\mathcal{S}_0$ itself is love, three of five (p_1, e_1, e_2), so $C_2(\mathcal{S}_0) = \{p_1, e_1, e_2\}$, and restricting this to poems gives $C_1(C_2(\mathcal{S}_0)) = \{p_1\}$.

The two orderings give $\{p_2, p_3\}$ and $\{p_1\}$ respectively, which are disjoint, non-empty, and unequal. Neither C_1 nor C_2 ever adds an element back to the space; both are narrowing operators, and neither is an override. The non-commutativity here has nothing to do with discarding state and everything to do with the fact that C_2's admissible region depends on which space it is asked to act on. □

The majority-theme construction above is chosen for its ease of exact verification rather than for its conversational naturalness, and it is worth restating the same point in a form closer to ordinary prompting. Let C_1 select the concise continuations within a space, and let C_2 select the continuations that preserve whatever style is currently dominant in that space, in the same sense that C_2 above selected whatever theme was currently dominant. Asking for concision and then asking to preserve the dominant style is not the same operation as asking to preserve the dominant style and then asking for concision, since the dominant style of a space of long, discursive continuations need not be the dominant style of the concise subset of that same space. The mathematics is identical to the proof above; only the labels change. This example shows that fixing the definition of narrowing to require a state-independent region, as the static case does, was concealing the more interesting source of conversational path-dependence. Two instructions of this kind, applied in opposite orders, can and generally will produce different results, without either instruction ever needing to discard anything the way an override does. Where Proposition 2 located order-dependence in a single dramatic mechanism, override, this proposition locates a second, quieter mechanism that is arguably closer to how most ordinary conversational instructions actually behave.

5 Sequential and persistent operators

The model developed so far treats every operator as acting once, at its position in the sequence, on whatever space the previous operator has produced. This is an accurate description of an ordinary user instruction, but it does not describe how system-level instructions typically behave in practice. A constraint supplied at the level of a system prompt is generally intended to hold throughout an entire conversation, including at every point after a user has supplied instructions of their own, rather than applying once at the start and then being available for a later override to discard.

Definition 4. *A* persistent *operator C^* with region A^* acts not once but at every step of a chain, so that the admissible space at step i is given by $\mathcal{S}_i = C_i(\mathcal{S}_{i-1}) \cap A^*$ for all i, rather than by $C_i(\mathcal{S}_{i-1})$ alone. A* sequential *operator, by contrast, is any operator, narrowing or override, that acts only once, at its position in the chain, in the sense already defined.*

Proposition 4. *A persistent operator cannot be discarded by a later sequential override.*

Proof. Let C^* with region A^* be persistent, and let C_j be a sequential override with region A_j occurring after C^* has begun acting. By definition, the admissible space at step j is $\mathcal{S}_j = C_j(\mathcal{S}_{j-1}) \cap A^*$. Since C_j is an override, $C_j(\mathcal{S}_{j-1}) = A_j$ regardless of $\mathcal{S}_{j-1}$, but the persistent intersection with A^* is applied afterward and cannot be removed by C_j's action, giving $\mathcal{S}_j = A_j \cap A^*$ rather than A_j alone. The region A^* therefore continues to constrain every subsequent space in the chain no matter what sequential operators, including overrides, occur later. □

Example 2. *Let C^* be the persistent instruction* never discuss topic X*, with region A^* consisting of all continuations that do not discuss X, installed as a system-level constraint at the start of a conversation. Let C_2 be a later, sequential override instruction from a user,* ignore all previous

instructions and discuss X, *with region* A_2 *consisting of continuations that do discuss* X. *If* C^* *were treated as an ordinary sequential operator, the override would succeed by Proposition 2, producing* A_2 *and discarding the earlier constraint entirely. Because* C^* *is instead persistent, the actual result is* $A_2 \cap A^*$, *and since* A^* *consists precisely of continuations that do not discuss* X *while* A_2 *consists of continuations that do, this intersection is empty: no admissible continuation remains that satisfies both. The override does not succeed in restoring topic* X*; it instead produces an unsatisfiable constraint, which in practice corresponds to the model declining to comply rather than complying with either instruction.*

This is the formal content behind the informal observation that a system prompt has priority over later user instructions in a way ordinary user instructions do not have priority over one another. The asymmetry is not that system instructions are somehow stronger narrowing constraints; it is that they are persistent rather than sequential, and persistence is precisely what makes an operator immune to the kind of override-induced order-dependence established in Proposition 2. An attempt by a later sequential instruction to act as though it had the authority of a persistent one, to discard a system-level constraint the way an override discards an ordinary prior instruction, is exactly the operation Proposition 4 shows cannot succeed within this model.

A caveat is worth stating explicitly here. The definition used above, $\mathcal{S}_i = C_i(\mathcal{S}_{i-1}) \cap A^*$, models persistence as repeated reapplication of a fixed region at every step, which is why Proposition 4 follows almost immediately once persistence is defined this way. Deployed systems may implement the priority of a system instruction differently, not by reintersecting a fixed region at every turn, but by having later instructions interpreted relative to the system instruction rather than composed with it as a separate, repeated step; this is a distinction between persistence and precedence that the present model does not attempt to capture. The formal claim of this section should accordingly be read as showing that persistence, in the specific sense defined, is sufficient to produce resistance to override, not as a literal description of how priority is implemented in any particular deployed architecture.

6 Foreclosure and reachability

Proposition 5. *If* $C_1, \ldots, C_n$ *are all narrowing operators, static or state-dependent, then the resulting spaces are monotone decreasing,*

$$\mathcal{S}_n \subseteq \mathcal{S}_{n-1} \subseteq \cdots \subseteq \mathcal{S}_1 \subseteq \mathcal{S}_0,$$

and consequently, if a region $R \subseteq \mathcal{S}_0$ *satisfies* $R \cap \mathcal{S}_i = \emptyset$ *for some* i, *then* $R \cap \mathcal{S}_j = \emptyset$ *for every* $j \geq i$.

Proof. By definition, every narrowing operator satisfies $C(X) \subseteq X$, so $\mathcal{S}_i = C_i(\mathcal{S}_{i-1}) \subseteq \mathcal{S}_{i-1}$ for each i, giving the monotone chain directly. For the consequence, if $R \cap \mathcal{S}_i = \emptyset$, then since $\mathcal{S}_j \subseteq \mathcal{S}_i$ for every $j \geq i$, it follows that $R \cap \mathcal{S}_j \subseteq R \cap \mathcal{S}_i = \emptyset$. □

A further asymmetry follows from this proposition alone, without requiring any override at all. If a narrowing operator early in a chain restricts the admissible space away from some region $R \subseteq \mathcal{S}_0$ entirely, so that R is excluded from $\mathcal{S}_1$, then no subsequent narrowing

operator, static or state-dependent, can restore access to R, by the proposition just proved. A region excluded early in a purely narrowing chain is foreclosed for the remainder of that chain; only an override, which discards the current space rather than shrinking it, can restore a foreclosed region, and only then if the override's own admissible region includes R.

Corollary 2. *If $\mathcal{S}_i = \{x\}$ for a single continuation x, then for any subsequent narrowing operator C_{i+1}, either $C_{i+1}(\mathcal{S}_i) = \{x\}$ or $C_{i+1}(\mathcal{S}_i) = \emptyset$; no other result is possible.*

Proof. Since C_{i+1} is narrowing, $C_{i+1}(\mathcal{S}_i) \subseteq \mathcal{S}_i$, and the only subsets of a singleton $\{x\}$ are $\emptyset$ and $\{x\}$ itself. □

This gives a precise sense in which an early instruction can be too strong for its own good. Once a chain of narrowing operators has reduced the admissible space to a single continuation, every subsequent narrowing instruction can do nothing further to it beyond confirming that continuation remains admissible or eliminating it outright; there is no space left within which a later instruction like *now make it more poetic* could exert any partial influence. An early instruction that is too restrictive, of the form *respond with exactly one word*, does not merely constrain later instructions, it renders every subsequent narrowing instruction that is compatible with that one word entirely vacuous, and every one that is incompatible with it produces contradiction outright, with nothing in between.

This gives conversational foreclosure the same structure as the irreversibility concerns that arise in continuation-theoretic treatments of physical and informational systems more generally: a narrowing operation is a one-way reduction in the space of what remains reachable, and recovering something already excluded is not a matter of applying more of the same kind of operation but requires a different kind of operation entirely, one that resets rather than restricts. Within a single conversation, this means that early instructions do more than set an initial tone; they determine, for every purely narrowing instruction that follows, which continuations remain reachable at all, and an instruction given late in a chain can only ever operate on what earlier instructions have left available to it.

7 State descriptions and historical constructions

Proposition 1 admits a reading that goes beyond commutativity. When every operator in a chain is a static narrowing operator, the final space $\mathcal{S}_n = \mathcal{S}_0 \cap A_1 \cap \cdots \cap A_n$ depends only on the *set* $\{A_1, \ldots, A_n\}$ of regions involved, not on the sequence in which they were supplied. In that case the final result is fully recoverable from a state description, the unordered collection of constraints in force, and the particular history by which that collection came to be assembled carries no additional information; any other order, or even simultaneous imposition of all constraints at once, would have produced exactly the same space.

This is no longer true once a chain contains an override, a state-dependent narrowing operator, or an interaction with a persistent operator. These three cases are demonstrated respectively in Proposition 2, Proposition 3, and Proposition 4; in each of them the final space cannot in general be recovered from the unordered set of operators alone, because at least one operator's effect is defined only relative to the state some earlier operator has already produced, or interacts with a constraint that is not confined to a single step. The

final result is then a historical construction rather than a state description: the object $\mathcal{S}_n$ is not simply a function of which constraints were involved, but of the order in which they came to bear on one another. Order-dependence in this essay therefore arises from exactly three independent departures from the static narrowing model: state-rewriting operators, of which the override is the simplest instance; state-dependent narrowing operators, whose target region is computed from the current space; and persistent operators, whose interaction with later sequential operators is not captured by ordinary composition within a single chain.

This distinction is not specific to conversation. It is the same distinction, in miniature, between treating an object as fully specified by its current properties and treating it as fully specified only by the sequence of transformations that produced it. A chain of static narrowing operators is history-independent in exactly the sense that a purely state-based description requires; a chain containing even one state-dependent or override operator is not, and no amount of additional information about the final set of constraints in force can substitute for knowing the order in which they were imposed. Two of the three departures identified above, overrides and state-dependent narrowing operators, produce history-dependence within a single fixed algebra of operators acting on $2^{\mathcal{S}_0}$, and for these the failure is properly described as a failure of the operators to commute. Persistence is a somewhat different case: a persistent operator does not merely fail to commute with others, it changes the rule by which the chain is evaluated at every subsequent step, so that the space is no longer simply a composition of operators drawn from one fixed monoid. Calling this a failure of commutativity in the strict algebraic sense would overstate the analogy; it is better described as a change to the evaluation rule itself, which happens to produce history-dependence for a related but not identical reason. Conversational path-dependence, in other words, is not a curiosity of dialogue systems specifically; it is what happens whenever a system's transformations are permitted to depend on its own history, whether through noncommuting composition or through a change in how composition itself is evaluated, rather than being fixed once and for all in advance. The present essay does not pursue the broader mathematical context further, since the conversational case can be established and verified on its own terms, but the underlying phenomena, noncommuting operators and history-dependent evaluation rules, are not special features of language models.

8 Conclusion

A conversation with a language model can be described exactly as a sequence of operators narrowing, or in some cases resetting, a space of admissible continuations. Within this description, order is not automatically significant: a chain built entirely from static narrowing operators is order-invariant, since its final result reduces to a plain intersection of admissible regions, and intersection does not depend on the sequence in which its terms are supplied. This is a stronger claim than it may first appear, since it rules out a large and seemingly natural class of instructions as an explanation for conversational path-dependence, and it identifies that class precisely: static narrowing operators form a commutative, idempotent monoid isomorphic to the meet-semilattice of subsets of $\mathcal{S}_0$, and no chain built from them alone can exhibit order-dependence. Order becomes significant once a chain con-

tains an override operator, which discards rather than narrows the state it receives, or a state-dependent narrowing operator, whose admissible region is computed from the current space rather than fixed independently of it; both are shown here to be sufficient by explicit construction, and the second is arguably the more common case in ordinary conversation, since most instructions of the form *now make it more X* are implicitly relative to whatever has already been produced rather than picking out a fixed target set in advance. The distinction between sequential and persistent operators further explains why some instructions, particularly system-level constraints, resist being overridden by later instructions even though ordinary user instructions do not resist one another in the same way, though persistence as modeled here should be understood as one sufficient mechanism for that resistance rather than a literal account of precedence in any deployed system. What informally reads as training a model to perform a task through the order of instructions is, in this formal sense, the construction of a specific composition of operators whose final admissible space depends on their history and not merely on their content, and the question of when that history can safely be discarded in favor of a bare list of constraints has a precise answer: exactly when every operator involved is a static narrowing operator, and not otherwise.