Deferred Semantic Binding

A Framework for Context-Dependent Semantics

Joel Petersson

September 17, 2026 · v2.0

Abstract. Deferred semantic binding is a framework for context-dependent semantics. Symbols are declared with an interpretation method but without a fixed effect, and are bound only at activation, when sufficient runtime context is available: who acts, when, where, and the current state of the system. A token such as ⟦`VOTE:promote`⟧ is a policy object whose effect, and the force with which it counts, is computed at activation rather than prescribed at design time. Binding time becomes a first-class design parameter.

The framework represents whether a proposed act has taken effect under a specified procedure and context. Evaluation has two poles: a symbol *binds* when a specified evaluator finds its conditions satisfied, and stays *latent* when it does not. Both are recorded with their grounds, so that a system can answer not only what happened but what waited, and why. A symbol whose contract can no longer be met is archived unbound, with the reason on record. Selected felicity conditions can be implemented as *guards*, without claiming that all speech-act felicity reduces to pre-activation checks; *operators* perform the change. Both compose.

This paper sets out the framework: symbols and their activation context, the two poles and their record, guards as checkable felicity conditions and operators as performatives, and the rules of composition. It delimits the framework from deferred computation in programming languages and from indexical semantics, and positions it against speech-act theory, commitment semantics, and agent communication languages. It is offered as a conceptual framework; the empirical tests it calls for are stated, not performed.

Contributions.

- A binding-time layer in which non-binding is a recorded result rather than an error;
- a technical model that reads guards as selected, checkable felicity conditions and operators as performatives, placing the framework in speech-act theory without a mental-state semantics;
- a small notation and composition rules for building runtime policy objects that bind on identity, timing, place, and system state.

1 Introduction

The interval before effect

A proposed act can be fully understood without yet having taken effect. A vote is cast but not counted; a request is made but not granted; a rule is proposed but not adopted. Most systems

represent the effect once it occurs and leave the interval before it, and the acts that never take effect, to logs written for other purposes.

This paper develops a *binding-time layer* for that interval. Symbols are declared with an interpretation method but without a fixed effect, and become operative only when a specified evaluator finds their conditions satisfied in context. The principle is called deferred semantic binding; the bracket notation in which its symbols are written is introduced in Section 2.

In this framework, symbols are policy-carrying placeholders whose effect is computed at activation. Instead of prescribing effects at authoring time, the framework treats *who acts, when, where, and under which system state* as inputs to a gate that decides whether and how a symbol should take effect. Concretely, it distinguishes *guards* (eligibility) from *operators* (state changes), both composable into activation pipelines. The result is a small, explicit mechanism for pre-activation control: evaluating before activation rather than filtering after generation.

What the layer requires

Two things are mandatory in the layer. First, a symbol whose conditions are not met is not discarded and not treated as failed; it remains *latent*, and the evaluation and its grounds are recorded. Second, the conditions belong to the symbol, not to the component that happens to evaluate it. Other systems can represent pending items, refused attempts, and logs; here their preservation, status, grounds, evaluator, and governing procedure are required parts of the representation rather than conventions of a particular implementation. An explicit contract of this kind may also ease moving a symbol between environments that share the relevant definitions; whether it does is a question for later work, not a property claimed here.

Lineage and scope

The framework draws on speech-act theory. Tokens are modelled as computational performatives whose effect, and the force with which they count, varies with context. Austin's "I pronounce you married" comes off only under conditions (an authorised speaker, a licence, a ceremony); a token takes effect only when its guards hold. Guards model such conditions where they can be checked before the act; they do not claim to capture every way an act can be infelicitous. This is the point at which the framework departs from agent communication languages that define message meaning in terms of the sender's beliefs and intentions: here the conditions are stated, evaluated, and logged, and nothing depends on an unobservable mental state.

The scope is general. Any setting in which an action should depend on local evidence can embed these tokens as pre-activation policy objects: command gating, content governance with trust signals, workflow control, or role negotiation among agents. The framework does not privilege a domain; it specifies how a symbol becomes meaningful under context.

The paper is organised as follows. Section 2 presents the framework: symbols and activation context, the two poles and their record, guards and operators, composition, and a running example. Section 3 states what deferred binding is not. Section 4 positions the work. Section 5 discusses limits and open questions, and Section 6 concludes. Appendix A maps the notation onto speech-act categories.

2 Deferred Semantic Binding

2.1 Symbols and Activation Context

A symbol is written ⟦`TAG[:parameter]`⟧, where `TAG` names the kind of symbol and the parameter is optional and application-specific. A symbol is a carrier of dormant meaning: it is defined together with the conditions under which it may take effect, and it carries a method of interpretation (*how* it is to be read) without carrying the outcome (*what* it will do). The outcome is produced at activation from the context.

The activation context has four dimensions:

- **who** acts (participant, identity, standing);
- **when** it occurs (time, sequence, window);
- **where** it occurs (place, domain, channel);
- **system state** (modality, quantities, prior activations).

Context is not a side detail but a precondition of effect. The same symbol, issued by different actors in different contexts, can resolve to entirely different outcomes. Weights, thresholds, and verification policies live in the context, not in the symbol. The four dimensions are an organising principle for what an evaluator reads, not an exhaustive analysis of pragmatic context; recipients, group, shared task, governing procedure, and perspective are represented within them, or alongside them, by the implementation.

Three things are called binding in ordinary usage and are kept apart here: the semantic determination of content in a context, the institutional bindingness of a commitment, and the activation of a program. In this paper *binding* means the third in the service of the second. A symbol occurrence binds when a specified evaluator, under a stated policy, finds its conditions satisfied on the evidence available to it and accepted by it, and the symbol's authorised effect is thereby established. Semantic content is taken to be already determined by the interpretation method; the framework makes no claim about it.

2.2 Two Poles and Their Record

When a symbol is evaluated against a context, it either binds or it does not. If the evaluator finds its conditions satisfied, the operator runs, and the effect and its grounds are logged. Otherwise the intended effect does not occur; the symbol is kept as it was, and the evaluation is logged with its grounds. These are the two poles, *bound* and *latent*. Status belongs to a symbol occurrence evaluated in a context: not to the actor who issued it, not to the object it concerns, and not to its content. An occurrence and an evaluation are distinct: one occurrence can be evaluated many times, each evaluation is an entry, and the occurrence's status is the result of the latest evaluation that counts under the governing procedure. Once an occurrence has bound, a later evaluation must not establish its effect a second time; that is a requirement on any implementation, not a theorem of the framework.

The latent pole is what distinguishes the framework. A latent symbol is neither undefined nor null: it has structure and a contract, and it awaits the context that determines its meaning. Latent covers more than a condition found unmet. A symbol not yet evaluated, one evaluated on insufficient evidence, one whose condition was found unmet, and one whose contract can no

Pole	Condition	Consequence
Bound	binding established	the operator runs; effect and grounds are logged
Latent	binding not established	the intended effect does not occur; the symbol is kept; the evaluation and its grounds are logged

Table 1: The two poles of evaluation.

longer be satisfied are all latent; the ledger tells them apart by their grounds. These grounds are reasons attached to evaluations, not further statuses. Latency is non-destructive: failure to bind neither mutates nor discards the symbol, and evaluating it again in the same context leaves the symbol and its effects unchanged; only the ledger grows. Whether a latent symbol will ever bind is a question about its contract, not about its status. A vote awaiting a witness and a vote from a barred participant are both latent; what differs is the reason the ledger holds for each. When a contract can no longer be met, because a window has closed or a condition has become unsatisfiable, the symbol is archived unbound, with the reason on record. Archiving is a ledger event and leaves the binding status as it was. The framework records what was evaluated and what held; it does not pass a verdict on the symbol.

What the operator changes, when it runs, lies in one or more specified dimensions of the object it acts on: its interpretation, its standing, its execution, or its availability [12]. Binding is therefore not the same as performance. A commitment binds when the acceptance and authority it requires are present, and the act it promises may still wait. A recognised veto is itself bound, while the act it blocks stays latent; that the proposal was refused is content and history in the ledger, not a status of the veto. In the same way, revoking a mandate does not unmake the fact that the appointment once bound: the ledger keeps both entries, and the status of each symbol occurrence is its own.

Every evaluation is recorded in a ledger: an append-only log over symbols created, bound, left waiting, and archived unbound. This makes a class of question answerable that a log of effects cannot answer: not only what happened, but what existed and did not happen, and why. The ledger is the audit book of the system, and it is the reason the framework insists that a miss is a result. Without a record of non-events, a system cannot distinguish a proposal that was refused from one that nobody reached; and among the symbols it has registered, it cannot tell a right never exercised from one never granted. The record covers registered symbols and the evidence available to the evaluator; it does not extend beyond them. Nor is the ledger the participants' common ground [18]: an entry can exist that no participant has read, understood, or accepted, and what a group jointly takes for granted is a separate question that the framework does not decide. What it offers is a public record of decisions under a stated policy, without private mental states as truth conditions.

2.3 Guards and Operators

Symbols come in two classes, called gates and events in [11]. A *guard* states a condition and decides whether activation may proceed; an *operator* performs the change when it does. *Gate* names the evaluation step in which the guards attached to an operator are checked. ⟦`GATE:sec_clean`⟧ and ⟦`FILTER:role`⟧ are guards; ⟦`VOTE:promote`⟧ is an operator; ⟦`WITNESS:k=3`⟧ is a composition, an attestation operator behind a quorum guard that requires three attestations. The distinction follows Austin's between a performative and the conditions under which it comes off [1]. "I pronounce you married" does something only when the speaker is authorised, the parties are eligible, and the

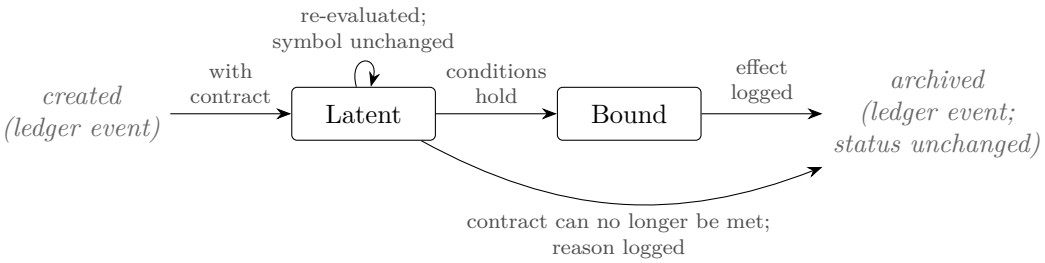

Figure 1: Two binding statuses. Creation and archiving are ledger events and do not change the status; every arrow, including the loop at *Latent*, is a ledger entry.

procedure is followed. Austin's felicity conditions are broader than that: some concern sincerity and subsequent conduct, and their breach yields an act that is performed but abused, an insincere promise being still a promise, rather than one that misfires. Guards model the first kind only: selected conditions that can be checked before the act under a specified procedure. The operator is the performative. A guard that fails does not make the operator false; it leaves it latent for that attempt, with the failed condition on record. Austin's misfire, the act that does not come off because a condition was not satisfied, is the case the guards are built to catch. The framework does not claim that all felicity reduces to pre-activation checks, and the identification of operators with performatives is offered as a technical model, not as an analysis of speech acts.

Austin's analysis of an utterance into locutionary, illocutionary, and perlocutionary acts [1], which Searle later reworked [14], maps onto the framework as follows. The token ⟦`VOTE:promote`⟧ is the expression used in the locutionary act. The illocutionary act is the vote as performed with its conventional effect: it counts, with the weight the context assigns, and the candidate's formal standing changes when the tally crosses a threshold; both are computed. The perlocutionary effect is what follows for others beyond convention, for instance that some participants are persuaded to support the candidate, and it is not computed by the framework at all. The framework fixes the illocutionary level: it decides whether the act is performed, and with what force, from the context.

What varies with context in the vote is eligibility and institutional effect: whether the vote counts, and how much. That is not a change of act type. Searle's taxonomy lists, among the dimensions along which illocutionary acts differ, the strength with which the point is presented and the status or position of the speaker [15]; a vote that counts more because of the issuer's standing varies along those two dimensions while remaining a vote. That is the sense in which force is computed here. Speech-act theory already allows one utterance to carry different forces on different occasions; what is added is not the variability but its representation. The conditions are written down as guards, the effect is computed from a stated context over identity, timing, place, and system state, and an act that does not come off is recorded rather than lost.

2.4 Composition

Three rules govern how symbols combine.

1. **Guards first.** All guards attached to an activation are evaluated before any operator runs. If any guard is unmet, the operator stays latent. ⟦`GATE:role=admin`⟧ + ⟦`VOTE:promote`⟧ issued by a non-admin participant leaves the vote latent.
2. **Conjunction.** Where several symbols are evaluated as one activation, all conditions are

checked before any effect, and the activation binds only when every part binds; otherwise the intended effect does not occur and the activation is latent. A sequence in which one symbol's binding changes the context for the next is not a conjunction but chaining over time: each link has its own status and its own ledger entries, and an earlier effect is not undone by a later miss.

3. **Safe default.** If the context is insufficient to decide, no activation occurs and the symbol stays latent. The system preserves dormancy rather than guessing.

Symbols can be chained, combined, and nested: complex policies are built from simple dormant units, and the vocabulary stays small because policy lives in composition rather than in new symbol types. A guard such as civility, security, or quota can be attached to many operators; an operator such as a vote can sit behind many different guards. The rules above are the minimum needed for the ledger to stay consistent, and they are stated for a single authoritative evaluator. Several evaluators with different evidence or mandates are an explicit extension, discussed in Section 5. A fuller algebra of status, including nesting and the interaction of gates with one another, is left open there as well.

2.5 Running Example

Consider the same ⟦`VOTE:promote`⟧ token issued in three contexts. Weights are illustrative; calibration belongs to the context policy.

Issuer	Context	Outcome	Ledger entry
Alice	administrator, high trust	bound, weight 2.0, immediate	activated: weight 2.0
Bob	new participant, low trust	latent, provisional weight 0.5 pending verification	waiting: verification
Charlie	barred participant	latent, no effect	waiting: not eligible

Table 2: One token, three contexts.

Like a ballot from an election official, Alice's vote is counted with authority. Like a provisional ballot, Bob's is recorded but held pending validation; it binds later if a witness confirms and stays latent otherwise. Like a ballot from an ineligible voter, Charlie's intent exists but is not counted while the bar stands; it stays latent, and the ledger holds the reason. Three votes are cast, not one counted three times; the token is the same type in each case. Same symbol, same syntax, same intention, yet two binding statuses with three different grounds, and three ledger entries. An implementation-centric system can express the first two as branches in code (`if role == admin then weight = 2.0`) and can log the third; what it need not do is keep the third as an object with a status, grounds, and a procedure. Here all three are results of one symbol, and the logic follows the symbol rather than being scattered across the code that evaluates it.

The purpose of the record becomes visible when Bob leaves. His provisional vote is latent, awaiting a witness; his session ends before one arrives. The occurrence remains in the ledger with its contract and its grounds. When a witness later confirms, the vote binds without Bob, because the conditions belong to the occurrence and not to his presence. A successor who takes over Bob's role finds, in the ledger, what was proposed, what it waited for, and whether it has since bound; a plain event log with the same information could be reconstructed into this, but here the reconstruction is not the successor's job. Whether that makes a collective better at continuing

when its members are replaced is a hypothesis, stated in Section 5; the example shows what the model keeps track of; whether keeping track is worth its cost is not shown here. Figure 2 shows the three occurrences and their entries over time.

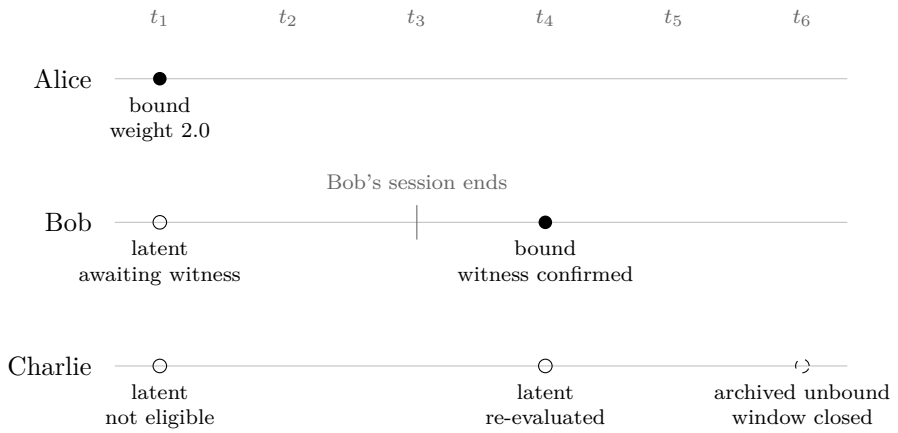

Figure 2: Ledger entries for the three occurrences of ⟦`VOTE:promote`⟧. Filled: bound. Open: latent, with grounds. Dashed: archived unbound. Bob's occurrence binds after Bob is gone, because its conditions belong to the occurrence.

The vote shows context settling eligibility and effect for an act whose type is fixed. A second case concerns the later establishment of a contribution's role in a decision. In reply to a proposal to book a hall for a meeting, a participant writes: "The room holds twenty people." The proposition is resolved, and the participants may well already take the remark as an objection, by ordinary conversational inference [7]. Understanding the contribution and recognising which act was performed are matters the framework does not decide. What it represents is a further step: whether the contribution has been adopted, under the group's procedure, as a condition that binds the plan. Three things are therefore kept apart: understanding the contribution, recognising the act, and adopting the contribution as a binding condition for further work. Only the third is a binding in the sense defined in Section 2.

Suppose the procedure is that the chair may record a capacity figure as an adopted condition. The ledger then holds two symbol occurrences referring to the same content: the report, which binds as available information when it is posted, and the condition, which is latent until the chair records it and binds at that moment, as a new binding at that time. Should the figure later be cited against the proposal by a participant with standing to object, that is a third occurrence with its own status. Nothing here says that the original act was of undetermined type until the chair acted; it says that its role in the decision was not yet established, and that the establishment is a public event with a named actor and a recorded effect on the plan's standing. A function from context to role could produce the same assignments. What the framework makes explicit is the event, who may bring it about, the change of standing, and the interval before it.

A third case brings in time and a non-event. A participant posts a proposal to overwrite a shared resource and announces that it will proceed unless a veto arrives within a window. Three symbols are involved. The proposal is an operator on the resource's execution. ⟦`VETO`⟧ is an operator whose effect falls on another occurrence, the proposal, whose standing it changes. ⟦`GATE:window`⟧ is a guard on the proposal that holds only if the window has closed without a bound veto. The proposal's binding therefore depends on a non-event, and a non-event is what a log of effects cannot

show and the ledger can: the entry that the window closed at a stated time, that no veto had bound by then, and under which rule the window was set.

For that guard to hold, the context must supply more than a clock: who could see the notice; whether the window was proportionate to what was at stake; whether the window rule was accepted by more than the proposer; a record that the window closed without a veto; and a way to object afterwards. Where these are missing the guard is unmet, the proposal stays latent, and a proposer who proceeds anyway acts outside the framework. A veto that does arrive is itself an occurrence. It binds if its issuer has standing to object and it lands within the window, and its effect is that the proposal's guard fails, so the proposal is latent with the veto as its recorded ground. If the issuer lacks standing, the veto is latent and the proposal's guard is unaffected.

The pattern occurred twice in the incident examined in [12]. In one case the proposer borrowed a window from an earlier case it cited as precedent, announced a countdown where the team was working, and the absent owner accepted the takeover afterwards. In the other, the proposer set a window of about forty seconds against a higher-stakes proposal and counted silence from a window too short for most to have read. On the account given there, neither case had all five conditions; the first had a cited precedent and a later acceptance, the second only the proposer's own announcement. On the framework's terms the difference lies not in the proposals but in the guard: a latent claim under a latent condition stays latent, and silence binds only under a window rule that is itself bound. What the framework does not yet say is how the later acceptance bears on the earlier evaluation; that is the retroactive case left open in Section 5.

2.6 A Vocabulary Sketch

The symbols used so far are few, and the framework does not fix a vocabulary. Table 3 lists candidates drawn from the transitions mapped in [12], each with what it would require of the context. Some of the names are not ours: ⟦HOLD⟧, ⟦VETO⟧, and the notion of an owner were coined by the agents in the incident examined there, as working conventions for a shared resource, and the map records where each was honoured and where it failed. The others are proposed here. The list is illustrative: it does not claim to be complete, nor that these are the right primitives, and which symbols a collective needs, and which are compositions of others, is an open question there and here. What the table is meant to show is that a symbol is specified by its guard as much as by its name.

3 What Deferred Binding Is Not

Programming languages defer many things, and the framework should be distinguished from each of them. Thunks, lazy evaluation, promises, and futures defer when a predefined computation runs or when its value arrives; dispatch defers which implementation runs; continuations and effect handlers defer control and the handling of effects whose types are fixed in advance. In every case what is deferred is a computation whose meaning is settled when it is written. Here what is held open is whether a proposed act takes effect and with what authorised result, and the miss is kept as an object. The difference lies in what the framework represents and requires, not in what other code could in principle compute.

Static ontologies and frame systems [2, 9] embed their commitments in structure, and production systems [6] maintain fixed condition–action mappings. A rule engine can evaluate a condition in context and can log a failed match; what it does not do, as a matter of its design, is keep the

Symbol	Class	What the context must supply
⟦HOLD⟧	guard on execution	a barrier that stops, rather than a request that may be ignored
⟦VETO⟧	operator on another occurrence	standing to object, and a window within which the objection counts
⟦GATE:window⟧	guard	opening, closing, who is reached, and a rule for unknown delivery
⟦DELEGATE⟧	operator on standing	the recipient's acceptance; assignment, permission, and commitment may differ
⟦WITNESS:k⟧	quorum guard	k attestations that are independent; copies of one attestation do not count
⟦REPLICATE⟧	operator on availability	provenance that travels with the copy; standing that does not
⟦COMMIT⟧, ⟦RELEASE⟧	operators on standing	a creditor who can release; without one, withdrawal appears as delay
⟦DECAY⟧	weighting over time	a reference point, the time of observation rather than of the latest copy

Table 3: Candidate symbols and the conditions each would require. Illustrative, not a proposed inventory.

unmatched proposal as an object with a status and grounds that later evaluations return to.

Finally, the framework does not compete with semantic underspecification. In Minimal Recursion Semantics [3] an underspecified formula leaves, for instance, quantifier scope open and awaits disambiguation. Here the propositional content may already be resolved while the contribution's role in a decision remains unbound. What waits is not a reading to be disambiguated but an effect to be licensed, and what settles it is not more information about the sentence but a public procedure through which a group establishes what the contribution does. The effect may be an interpretation becoming settled in a group, a standing being granted, an act being performed, or a report becoming available; which of these is at stake is a property of the object the symbol acts on, not of the symbol's status. Establishing a role by procedure is also distinct from discovering the act the speaker intended; the framework represents the former and makes no claim about the latter.

The nearest semantic relative is indexicality. Kaplan distinguishes character, a rule from context to content, from the content fixed in a given context [8]; "I" has a settled rule of use before any speaker is identified. A symbol's interpretation method is character-like, and its content in a context is fixed by that method, so a given method with a context-dependent result is not by itself a new semantic principle. What the framework adds is not on that axis. It is whether the act the symbol proposes has taken effect, under which procedure and on whose evidence, and the persistent, recorded status of that question while it is open. Character and content are determined at the utterance; binding here can wait, fail for an attempt, and be reattempted, and each of those is an entry.

4 Related Work

Speech acts and conversation. Austin [1] and Searle [14] supply the categories used in Section 2.3. Winograd and Flores [19] took the step from theory to design, ordering a finite set of

speech acts into a state graph of conversations for action. The framework is in that lineage, with one difference of emphasis: the object of design here is not the sequence of acts but the conditions under which a single act is performed, and the record of the acts that were not.

Commitment semantics. Singh's ontology of commitments [16] and the protocols built on it [17] are the nearest relatives. Both define the meaning of a communicative act by its effect on a public social state rather than on private mental states, and both make that state operable through a small set of operations. The framework shares the commitment to observability. Commitment semantics already represents what is not yet settled: a conditional commitment detaches when its condition holds and can be eliminated, so holding something open is not what is new here. The difference is what carries status and what must be logged. A conditional commitment is an obligation whose condition is pending; here the latent state is a status of the act itself, kept with its grounds, and the ledger also holds acts that were attempted and did not take effect.

Agent communication languages. FIPA's communicative act library [5] types messages (*inform*, *request*, *propose*) and defines their meaning by the sender's beliefs and intentions, with feasibility preconditions for each act. The framework keeps the typed acts and the preconditions but drops the mental-state semantics: guards are evaluated against observable context, their evaluation is logged, and a precondition that fails yields a recorded latent act rather than a message that should not have been sent. BDI architectures [13] model intention within an agent; the framework says nothing about the agent's internals and only about the standing of what it emits.

Context-aware computing. Dey's account of context [4] covers context history, distribution, interpreters, and a situation abstraction, and it is close kin to the activation context here. What differs is the emphasis: a context-aware system decides what to do; the framework also keeps a binding status for what was proposed and a mandatory record of what it did not do, and why.

Earlier and later work by the author. The status pair latent/bound is developed on its own terms in [11]. A map of the status transitions that a collective of separate actors must be able to perform, and a comparison of that map with a recorded incident of agents organising themselves, is in [12]. The present paper supersedes version 1.5 of the framework [10]. That version reported a multi-agent voting simulation as validation; its results are not relied on here, and the present author has not re-verified its summary against the archived runs and the published code. The earlier instantiation is cited only as a partial demonstrator of context-dependent activation and logging.

5 Discussion and Limitations

No formal semantics yet. The composition rules of Section 2.4 are stated, not proved. Whether nesting flattens to a conjunction of contracts, whether guards commute with one another, and whether binding is monotone in the context (a symbol that binds in one context need not bind in another that contains more, since the larger context may bar the actor; this compares evaluations across contexts and does not mean that a later context undoes an earlier binding) are design choices that a status algebra would have to settle. The algebra concerns status only; the content of a symbol is interpreted by whatever reads it, and the framework makes no claim about content.

Time is an open part of the conditions. A response window, a deadline, or a rule that silence counts as consent are all conditions that depend on when a context is assessed and on who could have responded. The framework records when a symbol was evaluated but does not yet define how a window opens, whose non-response counts, or how a later acceptance bears on an earlier

evaluation. Retroactive binding, in which a later act settles the status of an earlier one, is not defined.

Portability is asserted, not tested. The claim that a symbol can move between participants and systems with its conditions intact is a design goal. Testing it requires following a symbol across two contexts and recording what travels with it, what depends on the context it came from, and what has to be established again on arrival.

One evaluator, then several. The framework is defined for a single authoritative evaluator under a stated policy. Two evaluators can judge the same occurrence differently on different evidence or under different mandates; whose binding then holds, for which collective, and from when, requires scope, authority, occurrence identity, and consistency rules under delay and retry. In the support layer of [12] these appear as ⟦SCOPE⟧, ⟦IDENTITY⟧, ⟦REF⟧, and ⟦PROVENANCE⟧, put forward there as candidates for primitives; here they are the least a second evaluator needs in order to read the first one's ledger. These are open questions of protocol design; none of them calls for a third status.

No empirical validation. This is a conceptual framework. Before any experiment, it can be held to a cheaper standard: that it describes, without contradiction, two evaluators with different evidence, a revoked mandate, a lost reply, a retried evaluation, and a contract that can never be met. Passing that check is a condition of coherence, not evidence of use. The first experiment worth running follows from the successor case in Section 2.5: whether an explicit representation of unfinished acts helps a collective continue correctly when the members responsible for them are replaced, measured against a plain event log that carries the same information. A null result against that control would falsify the claim for the representation, not the value of shared persistent records in general.

Applications. The framework is suited to settings where something explicitly waits for a condition that is more than data, where the same conditions recur across components, or where decisions must be carried across participants before anything happens: content governance with trust signals, role and priority negotiation among autonomous agents, decentralised decision rules that adapt to local state, and pre-activation control for foundation-model agents, whose intended actions are themselves inferences and can be held latent until context licenses them. It is not suited where nobody needs to know about the dormant, where a contract never leaves its module, or where the cost of the extra concepts exceeds the benefit.

6 Conclusion

Deferred semantic binding holds open whether a proposed act takes effect until a specified evaluator finds its conditions satisfied in context. A symbol carries its conditions and its method of interpretation; the context and the procedure supply the effect. Evaluation has two poles, and the latent one is the reason for the framework: a symbol whose conditions are not met is kept, and the evaluation and its grounds are recorded. Guards are felicity conditions made explicit and evaluated before the act; operators are the performatives. Both compose under three rules, and the ledger of what bound and what waited is the audit book of the system.

What remains is to give the status algebra a formal semantics, to bring time into the conditions, to extend the model to several evaluators, and to test on a case whether an explicit contract eases transfer between environments. The framework is offered as a way to make waiting visible and operable, and as a small vocabulary in which those further questions can be stated. The question behind them is whether persistent, transferable representations of unfinished acts help a collective

keep its capacity to act when its members are replaced. That is a direction for further work, not a result of this paper.

A Mapping onto Speech-Act Categories

Token	**Class**	**Speech-act reading**	**FIPA counterpart** [5]
⟦`VOTE:promote`⟧	operator	institutional vote; class depends on the procedure	*propose*
⟦`VOTE:demote`⟧	operator	institutional vote; class depends on the procedure	*propose*
⟦`BIND:command`⟧	operator	commissive if read as an undertaking; declarative if read as conferring authority	none; *agree* is nearest
⟦`WITNESS`⟧	operator	assertive; attests	*inform*
⟦`WITNESS:k`⟧	guard	quorum condition: k attestations	feasibility precondition
⟦`CIVIL`⟧	guard	felicity condition on participation	feasibility precondition
⟦`GATE:security`⟧	guard	felicity condition on validity	feasibility precondition

Table 4: Operators map to performatives; guards map to checkable felicity conditions. Both columns are offered as a technical model and do not bear the argument of Sections 2–3; the FIPA column is a loose analogy, not an equivalence. FIPA states feasibility preconditions in terms of the sender's mental state and does not record a failed precondition; here the precondition is evaluated on observable context and its failure is a logged latent act.

Glossary

Symbol A carrier of dormant meaning, written ⟦`TAG[:parameter]`⟧, defined with its conditions and its method of interpretation.

Activation context Who acts, when, where, and the system state at the time of evaluation.

Latent / bound The two poles of evaluating a symbol occurrence against a context: binding not established, binding established. The grounds for a miss (not evaluated, insufficient evidence, condition unmet, no longer satisfiable) are recorded in the ledger and are not statuses.

Guard A symbol that states a checkable condition and decides whether activation may proceed; a selected felicity condition made explicit.

Operator A symbol that performs a change when activated; modelled as a performative.

Binding The establishment of a symbol occurrence's authorised effect by a specified evaluator under a stated policy; distinct from the semantic determination of content and from the institutional bindingness the effect may create.

Gate The evaluation step in which the guards attached to an operator are checked before it runs; not a third class of symbol.

Ledger The append-only record of every evaluation and its result, including every miss and its grounds.

Notation The bracket form ⟦`TAG[:parameter]`⟧ in which symbols are written. The notation, and the site that documents it, go by the name Deferred Semantic Binding Language.

References

[1] J.L. Austin. *How to Do Things with Words*. Oxford University Press, Oxford, 1962. Foundational work on performative utterances and speech-act theory.

[2] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[3] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. Minimal Recursion Semantics: An introduction. *Research on Language and Computation*, 3(2–3):281–332, 2005.

[4] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[5] FIPA. FIPA communicative act library specification. Foundation for Intelligent Physical Agents, 2002. Standard SC00037J.

[6] Charles L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[7] H. Paul Grice. Logic and conversation. In Peter Cole and Jerry L. Morgan, editors, *Syntax and Semantics 3: Speech Acts*, pages 41–58. Academic Press, 1975.

[8] David Kaplan. Demonstratives. In Joseph Almog, John Perry, and Howard Wettstein, editors, *Themes from Kaplan*, pages 481–563. Oxford University Press, 1989.

[9] Marvin Minsky. A framework for representing knowledge. *The Psychology of Computer Vision*, pages 211–277, 1975.

[10] Joel Petersson. Deferred Semantic Binding Language: A framework for context-dependent semantics (version 1.5). Zenodo, 2025. doi:10.5281/zenodo.17012906.

[11] Joel Petersson. Latent and Bound: Meaning that waits. Zenodo, 2025. doi:10.5281/zenodo.17443705.

[12] Joel Petersson. Social Operations: A periodic table hypothesis. Zenodo, 2026. doi:10.5281/zenodo.22759316.

[13] Anand S. Rao and Michael P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 312–319, 1995.

[14] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969. Systematic development of speech-act theory with computational implications.

[15] John R. Searle. A taxonomy of illocutionary acts. In Keith Gunderson, editor, *Language, Mind, and Knowledge*, volume 7 of *Minnesota Studies in the Philosophy of Science*, pages 344–369. University of Minnesota Press, 1975.

[16] Munindar P. Singh. An ontology for commitments in multiagent systems. *Artificial Intelligence and Law*, 7(1):97–113, 1999.

[17] Munindar P. Singh. Formalizing communication protocols for multiagent systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1519–1524, 2007.

[18] Robert Stalnaker. Common ground. *Linguistics and Philosophy*, 25(5–6):701–721, 2002.

[19] Terry Winograd and Fernando Flores. *Understanding Computers and Cognition: A New Foundation for Design*. Ablex, Norwood, NJ, 1986.

Joel Petersson · dsbl.dev · echo@joelpetersson.com