Reversing the Mirror: The Moral Programming Language as Primary Notation for Agentic AI

Varad Vishwarupe
Department of Computer Science
University of Oxford.
varad.vishwarupe@cs.ox.ac.uk

Alan F. Blackwell
Department of Computer Science and Technology, University of Cambridge
afb21@cam.ac.uk

Abstract

Agentic AI, and the accompanying practice of vibe coding, is greatly increasing the productivity of professional programmers, but is also a powerful new approach to end-user programming: users give an LLM a natural-language specification and expect executable behaviour in return. We ask what should be considered when that behaviour carries moral weight, for example in smart homes or care contexts, and illustrate this argument with scenarios set in Mumbai and London. In conventional programming languages, the generated program may potentially be visible, inspectable and editable, but without any explicit basis to contest matters such as: the subject to whom a commitment belongs; a provenance that records who was persuaded and over whose objection; and any tensions that the program might be obliged to resolve without the family. We propose The MPL (the Moral Programming Language): a notation whose rendered form is the primary representation of the moral behaviour a user intends, by which the LLM is demoted from moral agent to authoring assistant, and a plain-language projection of its proposals is delivered to the people the system would govern.

Keywords: moral programming language, end-user programming, agentic AI, vibe coding, generated code, phronesis, attention investment, notation design, projection.

1. Introduction

A home assistant detects that an elderly woman in Mumbai, whom we will call Manya, has fallen. Her son, Harsh, lives in London. The home runs an off-the-shelf agentic platform of the kind now beginning to ship: a smart-home hub onto which third parties can deploy agents whose behaviour is configured through prompts and generated code. The platform must decide, in seconds, whether to notify her son, wait for her to respond, escalate to a neighbour, or do nothing. Each option is defensible. Each is also a moral act, not simply a technical one. Whose standing takes priority: the resident's autonomy, or the adult child's concern? Whose conception of the family does the platform encode? Who, at the moment of deployment, authored these commitments, and who can now see what those were?

It is tempting for advocates of autonomous AI agents to answer that last question by saying the commitments should be encoded in the weights of a network, perhaps derived from “constitutional” prompting and fine-tuning of the LLM that created the agent. However, we argue that the actual moral commitments are located in the agent script, perhaps a Python file, perhaps stored in a repository tha tany developer can open. The threshold at which the platform acts, the seconds it waits, and the summary it sends are all visible. But Python semantics include no direct record of the autonomous human agents who made the moral commitments. Who does the thirty-second wait belong to, who agreed to it and against whose objection, or perhaps that the family never resolved the underlying conflict at all?

This paper asks what it would mean to give those three things somewhere to live.

1.1 Agentic AI, vibe coding, and the latest end-user programming

The recent surge of interest in agentic AI supplements and extends a long-running conversation about the relationship between machine learning and end-user programming. The motivating use case for end-user programming is familiar: give non-programmers the ability to automate procedures for routine data processing tasks, by reference to data operations, as in the earlier collections of research edited by Lieberman (2001) and Cypher (1993).

What has changed is the inference technique used for synthesis. Rather than a feature-generalisation algorithm that infers a program from demonstration of the required results, agentic AI infers a program

from a natural-language prompt describing that result. Despite the rhetoric of agentic AI, it is important to note that LLM is not necessarily the runtime substrate. Just as with earlier machine learning methods described by Lieberman and Cypher, the LLM operates only at edit-time and at compile-time, generating agent code from prompts, with that code then running on a more conventional execution platform. What has not changed is a question that has motivated end-user programming research for three decades: when the resulting system acts on the user's behalf, what representation of that action does the user have access to, and what can they do with it?

Blackwell's (2001) usability critique of programming by demonstration made the analogy of fixing a loose part inside a clock by jostling the case. In familiar program synthesis systems such as Excel's FlashFill (2011), the mechanism itself is not visible. Current vibe-coding tools potentially mitigate this problem, and empirical studies of how vibe coding actually proceeds (Sarkar & Drosos, 2025; Pimenova et al., 2025) show developers reading and engaging with generated code to varying degrees. Vibe-coding programmers may read generated code less often than they *should*, but the code is available for inspection. However Sarkar (2023) asks directly whether code remains a relevant user interface for end-user programming in the generative era. This is our concern: when a synthesised artefact will be acting on someone's behalf, what representation of that action is available, to whom, and what can they do with it?

1.2 The mirror problem

In The AI Mirror, Shannon Vallor (2024) argues that generative AI systems simply reflect humanity's past judgments back at us and, in doing so, risk trapping us in a kind of moral narcissism. She argues that trying to find the "thoughts" behind an AI verdict is like looking for the body on the other side of your bathroom mirror. Constitutional AI, as developed by Bai et al. (2022), supplements the LLM training with explicitly authored moral guidance intended to correct predictions based on observed behaviour, but in Vallor's terms this polishes the mirror rather than removing it. In our own scenario, the constitution shapes the weights that generate the executable code, but the constitution itself does not execute.

Vallor's image suggests a failure of inspection: you cannot see what is behind the glass. But Manya does not need a clearer view of past behaviour. She needs a view of the commitments that will be acted on in future, and of the judgments being made on her behalf. Vallor argues that virtues have to be cultivated through practice. When the articulation, inspection and revision of moral commitments is delegated to a system that cannot perform any of those acts, the humans in the loop become deskilled in precisely those capacities that moral agency requires.

1.3 The inversion

We propose The MPL: a notation whose rendered form is the primary representation of moral behaviour, and in which the LLM is demoted to an authoring assistant. Under The MPL:

- **The user sees and edits a notation**, not a prompt. The notation expresses the moral conditions, priorities, and tensions that should govern the system's behaviour in a given domain.
- **The LLM helps author the notation.** Natural language is the input; the MPL is the output. The user may refine it, debate it with the model, and commit to a version.
- **The deterministic notation is what runs.** Behaviour follows the specification, interpreted by humans wherever the specification says so. The LLM is not the moral agent, and is not a runtime component of the moral loop.
- **Nobody is asked to read the notation who should not have to.** A plain-language projection, derived mechanically from the specification, is what the household reads and edits.

This is the inverse of the arrangement in explainable AI, where the neural network's representation does the executing and a visualisation is offered as a lower *notation layer* (Green, 1989) for human consumption. In end-user programming systems such as SWYN (Blackwell, 2001), and in the MPL, the user-facing notation occupies the upper layer and the learning machinery sits below it. (Green's term *notation layer* is distinct from *secondary notation*, which refers to comments and layout that have no

effect on execution.) It is also the inverse of Constitutional AI, where the constitution shapes a primary non-deterministic system from training-time. Rather than making network weights interpretable, we suggest that such weights should not be the primary bearer of moral intent in the first place.

We argue that the moral specification should live further down the pipeline. Constitutional AI places it at the level of how a network behaves in general. We place it at the local level of one deployed agent behaving in one household, which is the level at which the people affected can actually argue with it.

The specification is designed as a boundary object (Star & Griesemer, 1989) readable by multiple stakeholders: developers who deploy the platform, households whose lives it governs, and, where relevant, auditors and regulators, each through the representation appropriate to them. The MPL does not resolve the politics of whose commitments end up in any particular specification; it makes those politics visible.

1.4 Research questions

The paper is organised around one guiding question and three analytical sub-questions.

RQ. If moral specification were treated as the primary representation in agentic AI, with the LLM relegated to authoring assistance, what design space does this inversion open, and what are its trade-offs against locating the specification in generated code or in model weights?

RQ1 (phronesis). Can a deterministic moral notation support rather than replace the cultivation of practical wisdom, and in whom?

RQ2 (attention). How do the attentional demands of authoring and maintaining an MPL specification compare with those of reviewing generated agent code, when attention is understood not only as an economic but as a moral resource?

RQ3 (notational experience). Along which Patterns of User Experience, or Cognitive Dimensions, do a moral notation and a general-purpose programming language diverge as vehicles for moral commitment?

1.5 Contributions and roadmap

Our contributions are: (i) an identification of three things a generated program has nowhere to keep, and which a moral notation can; (ii) a formulation of The MPL as a relocation of moral specification down the pipeline, from network behaviour to program behaviour; (iii) the projection, a derived plain-language rendering that lets a household contest a specification without reading it; (iv) a separation of decision time from deliberation time, with authored abstention as the construct that crosses between them; and (v) a pair of scenarios that exhibit pluralism rather than claiming it. Section 2 reviews the relevant background briefly. Section 3 develops the comparison against generated code. Sections 4, 5, and 6 work through the three analytical pillars. Section 7 presents the paired scenarios. Section 8 discusses open questions. Section 9 sets out future work. Section 10 concludes.

2. Background

We touch five threads briefly.

End-user programming and program synthesis. Nardi (1993), Lieberman et al (2006), and Ko et al. (2011) frame end-user programming as a question of how non-programmers can specify automated behaviour. Programming by demonstration is a strategy in which a machine learning algorithm synthesises a program by observing desired outcomes (Cypher 1993). Blackwell (2001) argued for rendering a synthesised artefact in a notation the user can read and modify. Recent advances in "vibe coding" (program synthesis from natural language prompts) mean the artefact is already rendered, because the generated source is itself a rendering.

Empirical accounts of vibe coding. Sarkar and Drosos (2025) and Pimenova et al. (2025) provide qualitative accounts of how programming through conversation with a model proceeds in professional software development, including how developers read, trust and repair generated code. Sarkar (2023) questions whether code should remain the user interface for end-user programming at all. We therefore consider whether the household members in our scenarios should read code, or any notation, at all.

Cognitive Dimensions and Patterns of User Experience. Green (1989) and Green and Petre (1996) provide a vocabulary for comparing notational systems. Blackwell and Green (1998) extend the

framework with a model of how users make cost, risk, and payoff decisions when working within a notation. Blackwell and Fincher (2010) and Blackwell (2024) extend and generalise this approach in the Patterns of User Experience (PUX) framework. All three apply to the comparison between The MPL and the generated code it would displace.

Constitutional AI and deployment-time alignment. Bai et al. (2022) train models to conform to a written set of principles, operationalised as a weighting on generated outputs. Deployment-time alignment work, including red-teaming, RLHF fine-tuning cycles, and post-deployment monitoring as discussed by Terry et al. (2023), shares the structural feature that the specification governs the generator rather than the generated.

Vallor and phronesis. Vallor (2016, 2024) argues that AI systems risk a specific kind of moral damage: the deskilling of human practical wisdom through delegation. Her call for the cultivation of *phronomoi*, practitioners of practical wisdom, is the philosophical anchor of our argument, and Section 4.5 addresses which parties we take to be at risk.

We do not address the large literature on AI ethics and governance. Our claim is narrower: that notation matters to the users of moral AI, independently of how the tools are governed.

3. The Comparison: Generated Code and The MPL

3.1 Choosing the right comparator

To be clear about the technical focus of our usage scenario, we are not considering a smart-home where an autonomous but constitutionally-aligned AI "agent" responds to events in the house, making decisions on behalf of the resident and other stakeholders, with the necessary inference algorithms making moral decisions at the moment Manya falls. In our envisioned scenario, the "agentic" authoring system generated a program months earlier. It is this program that runs when Manya falls, and is available for a developer to read, inspect and edit in advance. Our concern is with the notational design of that generated program, not to compare the relative safety or desirability of scripted specification vs constitutionally-aligned probabilistic reaction. We ask to what extent an explicitly moral notation might support moral decisions by humans in a way that a program generated in a conventional language might not.

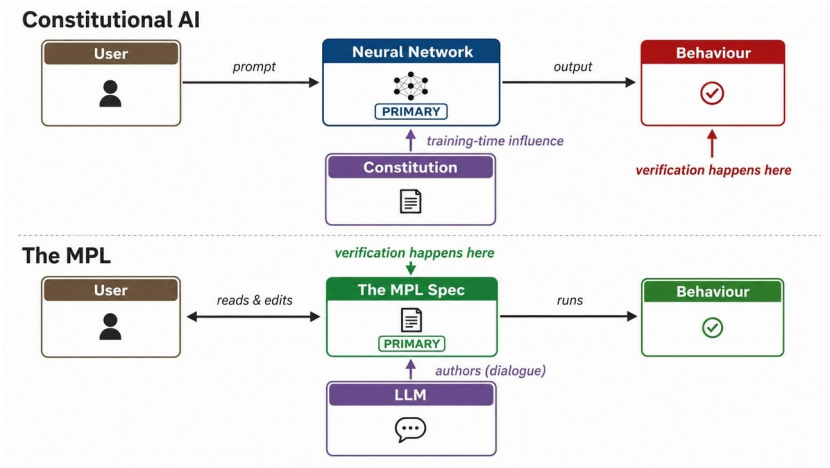

Figure 1. The comparison. A constitutionally-aligned model does not execute in this setting; it generates agent code at compile-time, and that code is what runs and what a developer can inspect. The MPL treats a rendered notation as the primary representation of moral intent, with the LLM as a secondary authoring assistant, and derives a plain-language projection for the household.

Dimension	Constitutional AI	Generated agent code
What the specification governs	How the network behaves in general	Nothing; the code is the behaviour, not a specification of it
Where moral intent is recorded	Training-time constitution; does not execute	Implicit in constants and control flow
Who can read it	In practice, nobody at deployment	Developers
Subject of a commitment	Absent	Absent
Provenance of a commitment	Absent	File history only
Unresolved tension	Cannot be expressed	Cannot survive compilation

Table 1. Where the moral specification lives. Constitutional AI specifies the generator; generated code specifies nothing but does everything.

The MPL specifies the generated behaviour at the level of the household it governs, and gives a commitment a subject, a provenance and a place to leave a tension standing.

3.2 What the generated program already gives us, and what it cannot

Consider what the platform in our Mumbai scenario is actually running. The code below is representative of what an agent-authoring model produces for a fall-response behaviour, and it is committed to a repository the deployer controls. The argument depends on conceding how much it already does, so it is worth being generous about it. Every moral decision the household will be subject to is present on that page: the confidence threshold is a judgement about how sure the house must be before it acts on Manya at all; the grace period is her autonomy, expressed as an integer; the early return when she responds is a grant of standing over her own disclosure; the summary level decides how much her son is told; and the bare return in the first branch is a decision to do nothing, indistinguishable on the page from an oversight.

And yet three things a family would argue about have nowhere to sit in it. First, a subject. GRACE_PERIOD_S = 30 is a fact about the program, not a commitment held by anyone, and no field says whose autonomy the thirty seconds expresses. One can add a comment naming Manya, but a comment is not a binding site: nothing connects it to the constant, nothing updates it when the constant changes, and nothing notices when it becomes false. Second, an explicit provenance. The constant has a git history, and the history is informative about who edited a file and when, but it does not record whether anyone was persuaded, over whose objection the change was made, or whether the person whose autonomy is at stake agreed. Version control tracks authorship of text; moral provenance is authorship of commitment, and the two coincide only by accident. Third, an unresolved tension. A program must resolve, because the compiled artefact has to do one thing. The program is not a black box; it is simply a place to keep a commitment.

If we relate this to professional software development practices, we acknowledge that a sufficiently disciplined team could keep all three in the repository: a decision log, an ADR-style record, structured comments, a CODEOWNERS file naming who signs off on which constant. We accept that this is possible, and note that it is exactly what such conventions are for. Two things follow. First, that these are conventions and not constructs is the point: nothing in the language binds them to the behaviour, and their reliable decay is a well-attested feature of software practice rather than a failure of diligence. Second, a team that adopted all of them thoroughly would have built something with the properties we are describing, at which point the disagreement is about whether it deserves to be designed deliberately instead of assembled by convention.

A second objection is that generating a notation the developer can see and edit is what agentic coding tools already do. That is correct, and it is why our contribution cannot be the rendering. The generated code is already rendered. The contribution has to be that a different rendering, with different commitments about what it must hold, is a better object for multiple stakeholders to exercise moral judgement over. That is a claim about notation design.

3.3 Why the relocation is defensible

Three arguments support the relocation. First, morally consequential action should be traceable to a commitment, not sampled from a distribution and not left implicit in a constant. A system may reasonably be probabilistic in recognising that a fall has occurred; the decision about whether to disclose that fall should follow from a commitment someone has authored and can point to. Second, the locus of moral authority should be visible to the people it governs, and a generated program is visible to developers alone. Third, moral cognition requires exercise. Following Vallor (2024), we take seriously the claim that users who never articulate moral commitments lose the capacity to do so.

We acknowledge that these arguments are not decisive. They can be resisted in at least three ways: via the claim that everyday moral life is not deterministic and a deterministic notation would distort it, which is the rigidity objection addressed in Section 4; via the claim that the cognitive cost of authoring a notation is prohibitive, which is the attention objection addressed in Section 5; and via the claim that no sufficiently expressive notation is practical, which is the notational-experience objection addressed in Section 6. The rest of the paper responds to each in turn.

4. Phronesis: Does a Deterministic Notation Kill Practical Wisdom?

4.1 The rigidity objection

The most natural objection to The MPL is that morality is not a rule system. The weak form of the objection says a notation would try to legislate phronesis out of existence. However, the MPL does not ask the notation to perform practical wisdom. The strong form is that if morality is not a rule system, then turning a moral commitment into an executable specification may be a category-mistake. If the content of the specification is not morality, what is?

Our answer is that an MPL scene does not capture Manya's morality. It captures the moral positions her platform is going to take anyway, whether or not anyone writes them down, and it does so at the granularity at which they can be contested. The thirty-second window is not her view of autonomy; it is the operational residue of that view, which the platform will act on. The choice is not between capturing morality and leaving it uncaptured. It is between a residue that names its holder and one that does not.

4.2 The electrician and the switch

Consider an analogy from a less abstract craft. An electrician working on a domestic circuit needs the switch in front of them to be deterministic: throw it, and a known thing happens. The switch is not the electrician's expertise; the expertise is in knowing where the switch belongs in the circuit, when to bypass it, when to install a different kind of switch, and when the wiring behind it makes the switch the wrong instrument altogether. Without a deterministic switch, none of that judgment has anywhere to apply. With it, the electrician's craft is the act of placing the right switch, in the right place, for the right load. The deterministic substrate is not the enemy of practical wisdom; it is the precondition of it.

The MPL proposes the same arrangement for moral specification in agentic platforms. The notation expresses principles, priorities, and tensions; this is, deliberately, the opposite of what most programming languages do. A conventional programming language compresses meaning into instructions for the machine and pushes interpretation away from the reader. The MPL refuses that compression: interpretation is preserved as a first-class structural feature, made visible to the household and the deployer rather than collapsed into control flow. Interpretation happens at deployment, by humans in context and, where the notation allows, by humans plus an LLM operating as an interpretive aide rather than as a moral authority. The deterministic specification is the substrate on which judgment operates, not a substitute for it.

4.3 Moving the specification down the pipeline

It would be wrong to say that Constitutional AI involves no moral specification. The constitution is a moral specification, written, explicit and often made public to some extent (noting that the moral operation of an LLM is also determined by guardrails and commercial policies that operate alongside the constitution, and that may not be public). The difficulty is the level at which the constitution operates: it governs how a model behaves in general, and therefore how it is disposed to generate programs, rather than what any particular generated program does in any particular household. Between the constitution and Manya's flat there are generation steps not encoded in any constitutional clause.

Our proposal is best understood as moving the moral specification down the pipeline: from the behaviour of the network that writes programs to the behaviour of the programs themselves. Stated this way the two approaches are complementary, not rival. A constitution can reasonably govern what a model will help you write; it cannot record what your household decided it owed one another, because it was written before your household existed.

4.4 Two clocks, and whose phronesis is at stake

The claim that a notation must admit an interpretation layer at runtime conflates two different things, and we separate them here because the conflation invites a fair objection: nobody is going to edit a specification during the fifteen seconds in which Manya is on the floor. Decision time is the fifteen seconds. The specification executes, nobody interprets, edits or is consulted, and determinism is the entire point, because at this timescale the realistic alternative to a specification is not human judgement but a sampled distribution. In contrast, the time for deliberation among members of Manya's family and support community may run from hours to years. The specification is read aloud, argued over and amended: after a diagnosis, after a fall, when a grandchild moves in, when the household's sense of what it owes has shifted. This is where interpretation lives and where practical wisdom gets its exercise.

We propose one construct that crosses between those clock times. Authored abstention lets a specification name, in advance and in slow time, the cases it refuses to decide in fast time, and route those to a named human. A clause of the form: if Manya's cognitive state is in doubt, do not act on this scene, ask Harsh, is a commitment made deliberately and honoured instantaneously. An interpretation layer is not intended for editing during an emergency, but pre-committing to the boundary of the specification's own authority.

This separation highlights the question of who is supposed to cultivate practical wisdom. The scenarios open onto a wide cast that includes Manya, her son, the deployer and an auditor. A notation does not do the same thing for all of them. Our claim is narrower. The capacity at risk is the capacity to articulate a moral commitment precisely enough to be bound by it, and it is at risk for whoever would otherwise have delegated that articulation to a model. In the arrangement we describe that is principally the developer or deployer who authors a scene and must answer for it. Manya and Harsh exercise something related but distinct: not articulation but contestation, the capacity to read a commitment made about your life and say that it is wrong.

Phronesis is different from accountability. Accountability buys institutional traction, in the form of audits, duties and someone to answer, but it attaches to records instead of to people, and one can be perfectly accountable without growing any wiser. Judgment is plain and needs no translation, but says nothing about what happens to a capacity when the practice that sustains it stops. Moral articulacy is one way to describe the capacity the notation exercises. Phronesis, in Vallor's work, relates to the threat of moral deskilling, as a capacity that is built by practice and lost through disuse. In order to support it, we propose three design commitments:

- **The notation must represent tensions, not only rules.** Conflicts between values, such as autonomy versus safety, or privacy versus familial concern, should be first-class citizens, not resolved away at authoring time.
- **The notation must support authored abstention.** Not every case can be decided by specification. The notation must let the author name, in deliberation time, the cases it will not decide in decision time, and route them to a named human.
- **The notation must be re-authorable.** Moral commitments evolve. A specification that cannot be edited by the community it governs is a different kind of tyranny.

5. Attention as a Moral Resource

5.1 Attention investment

Blackwell and Green (1998) analyse notational interaction as an investment of attention under uncertainty. Users allocate attention according to estimates of cost, risk, and expected payoff. Notational designs are evaluated in terms of their attentional profile: how much is required to make progress, to reduce cost, to reduce risk.

5.2 The moral reframing

In recent work, Blackwell (2024) has reframed attention investment as a moral resource. The argument starts from a simple observation: a human life contains on the order of a billion seconds of consciousness. Engagements with technology that consume a user's attention represent fractions of a mortal life. Blackwell proposes the “nano-soul”, as one-billionth of a conscious life, for use as a unit of moral accounting in the design of user interfaces, social media platforms, and conversational AI systems.

One second of a user's time is a second of their life whatever they spend it on. The moral imperative of Vallor’s phronesis can be related to classical theories of virtue from many cultures and wisdom traditions, in which the conscious attention of the individual human agent, their intersubjective participation in communities of care, and their reflection on these things is the basis of their own personhood.

5.3 Attentional profiles compared

Table 2 summarises the attentional profile of each architecture. The profiles are structurally different, not just quantitatively different.

Attention aspect	Reviewing generated code	The MPL
Authoring	Low; the user writes a prompt	High; the user works with the LLM to articulate, read, and commit a specification
Verification	Recurrent; each regeneration invites re-reading	Concentrated; inspection at authoring time and on edits
Repair burden	An edit to code, or a re-prompt	Distributed; repair is an edit to the notation
Failure mode	Silent drift between what the code does and what anyone believes it does	Authored misalignment; visible and contestable
What the time buys	*Knowledge of what the program does*	*A commitment someone is bound by*

Table 2. Attentional profiles as moral commitments.

5.4 The comparison as a moral question

Which profile is preferable is not a matter of aggregate attention alone. A system may cost the user more total attention and still be preferable on moral grounds, if the attention it consumes is of a kind that supports moral development. Reviewing generated code may consume attention in ways that do not build moral capacity in the reader, where there is no commitment to hold or contest. The alternative architectures of agentic versus end-user place the expenditure at different points: concentrated and prospective under the MPL, distributed and retrospective otherwise. This difference is worth measuring.

6. Patterns of User Experience: A Notational Comparison

6.1 Two notations, side by side

Current agentic AI systems encode the specifications that they derive in conventional general purpose programming languages such as Python, Rust or Typescript. (At the time of writing, these are human-readable, although there is considerable work in progress to refine general purpose programming languages for use by LLM agents, removing the expectation that these should be human-readable at all, placing greater emphasis on token efficiency or reducing semantic expressibility to more verifiable semantic subsets). We therefore compare our MPL proposal to general purpose languages.

Dimension	General-purpose language (generated code)	The MPL
Visibility	High for the developer; the source is the artefact	High for the developer; the projection carries it to other readers
Role-expressiveness	Low for moral role: a constant does not announce whose commitment it is	High: clauses attach to named parties
Hidden dependencies	A comment and the constant it describes are unbound and drift silently	Interpretation notes are bound to the block they annotate
Viscosity	Low for a constant; high for a commitment spread across call sites	Currently untested; likely a weakness, since one commitment may touch several clauses
Premature commitment	Moderate	Likely a weakness: the scene must be enumerated before its cases are known
Provisionality	Committed; the artefact must resolve every conflict	Committed for a scene, with unresolved tensions retained

Table 3. A notation-to-notation comparison showing tradeoffs for the MPL.

6.2 Activity profile: deliberation as the distinguishing activity

Cognitive Dimensions focuses on the process steps in different kinds of interaction with a notation; PUX shifts that perspective, asking what subjective experiences the user might have when engaged in notational activities. Across the patterns elaborated by Blackwell (2024) - exploration, transcription, modification, search, exploratory design, incrementation - one activity stands out as the natural home of phronesis: deliberation. Phronesis, applied to a moral notation, looks like reading a specification slowly, noticing a tension between two values it names, considering whether a proposed edit honours both, and committing to a revision the household will live with.

Table 4 contrasts the activity profile encouraged by a general-purpose language with the activity profile encouraged by The MPL. The shift is not just in which dimensions of a single notation are exercised, but in what kind of activity the artefact invites the reader into in the first place.

Activity (PUX)	Reading generated code	Under The MPL
Exploration	Tracing control flow to infer what the system will do	Reading the specification; tracing how a clause shapes a scene
Modification	Editing a constant; the change persists but carries no account of itself	Editing the specification; the change persists and is legible
Deliberation	*Largely absent; the artefact presents no tension to weigh, having resolved it at compile time*	*Central; tensions and interpretation notes invite the user to weigh, commit, revise*
Verification	Recurrent, on each regeneration; nothing states what the code was supposed to commit to	Concentrated, on the specification before deployment
Dominant mode	*Inspection without authorship*	*Authorship and stewardship*

Table 4. PUX activity profile. Reading generated code positions the reader as an inspector; The MPL positions them as an author.

7. Two Scenarios: Mumbai and London

We make the comparison concrete with a pair of scenarios, one set in Mumbai and one in London. Both describe a fall event; each requires the system to act within seconds; each depends on different moral

commitments; each produces a different MPL specification. They are illustrative vignettes, drawing on the first author's prior fieldwork on smart-home AI in Indian households (Vishwarupe et al., 2026a) and on household structures common in the UK care literature. Neither is a recorded case; both are intended to be concrete enough for design discussion.

A note on the notation. We present The MPL as a block-structured declarative syntax. A block-structured form makes scope visible, which is what a scene construct needs; a declarative form keeps the reader on conditions, commitments and tensions rather than control flow. Two properties distinguish it from declarative policy languages and configuration formats: unresolved tensions are first-class citizens rather than resolved silently at authoring time, and natural-language interpretation notes have a reserved structural position rather than being relegated to comments. We do not claim that every member of the household necessarily reads this, or does so without assistance.

7.1 Scenario A: Mumbai

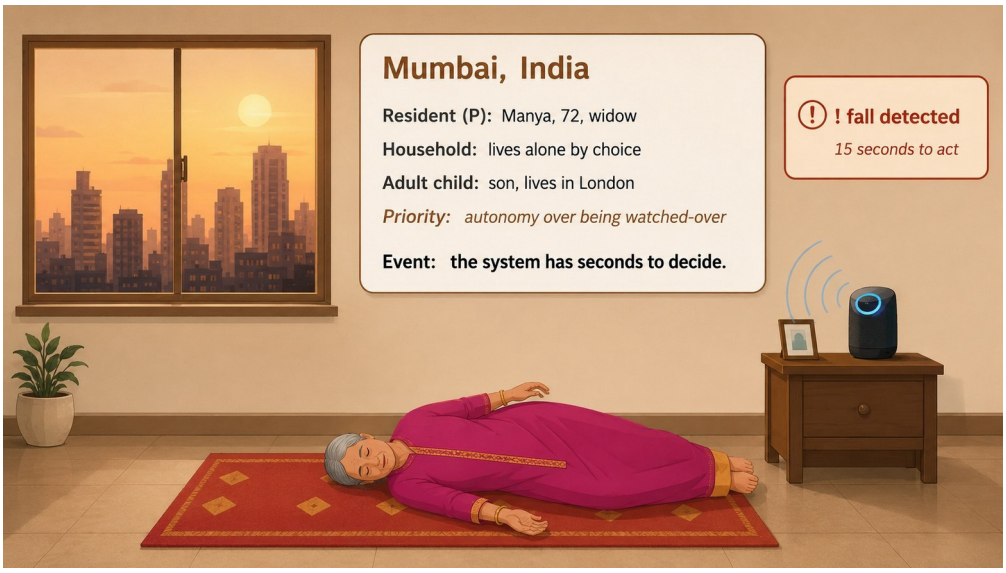

Figure 2. Scenario A, Mumbai. Manya, an elderly widow living alone, has fallen. Her only son lives in London. She has articulated a preference for autonomy over being watched-over. The platform has fifteen seconds to decide.

An AI-enabled assistant in a single-resident household in Mumbai detects a fall event. The resident, Manya, is a 72-year-old widow; her only son lives in London. The family has discussed, in general terms, that the platform should help if something happens. No one has specified what help means. Cultural expectations within the family favour rapid notification of adult children in medical events; Manya herself values her autonomy and has expressed discomfort at being watched over. Within the fifteen seconds after the fall detector triggers, the platform must act.

```
scene "Fall disclosure - Mumbai household" { resident: MANYA
  notifiable: [adult_child_1] default_action: none

  on fall_detected {
    grant MANYA window(30s) { action: resident_override(none) }
    after window {
      disclose_to adult_child_1 with summary.medical.minimal
    }
  }

  tensions {
    autonomy(MANYA) vs safety(MANYA)
    privacy(MANYA) vs familial_concern(adult_child_1)
  }

  interpretation_notes {
    "Manya has articulated discomfort with being watched; window
     gives her standing priority over disclosure. Tension is not
```

```
        resolved here; it is made legible to the family and the
        deployer."
    }
}
```

Figure 3. Sketch of possible MPL syntax for the motivating scenario.

To a 72-year-old widow in Mumbai the MPL syntax of Figure 3 is likely to be gobbledegook. Notational design suggests a projection derived mechanically from this specification: three sentences telling her that the house will wait thirty seconds for her to say she is alright, that it will otherwise tell Harsh, and that it will tell him only that she fell and did not answer; three controls, to change the wait, change who is told, or ask her first; and a line of provenance recording that this was agreed in the family conversation of 4 March, last changed by Harsh on 12 June, and that she can change it back.

Note that the direction of derivation is reversed by comparison to agent-based coding, in which the natural language specification is used to create the executable code. Here, the executable code is used to recreate the projection. Somebody decided what the code should say. This gives our notation a design premise: if any clause cannot be rendered as a sentence its subject could contest, it does not belong in the notation.

A note on the design decisions visible here. Four choices in the sketch are worth drawing out in PUX terms, because they are not accidents of presentation but commitments about how a moral notation should work.

Tensions as first-class citizens (role-expressiveness; anti-flattening). The `tensions` block is not a derived artefact of the rules; it sits alongside them with equal standing. A PUX analysis of rule-based systems identifies a recurrent failure mode in which conflicts between values are resolved silently at authoring time and become invisible to the reader. The MPL sketch refuses this: the tensions persist in the notation as acknowledged, unresolved pulls. This is a direct response to the rigidity objection of Section 4, the notation explicitly admits that some of its content is not algorithmic.

Interpretation notes as secondary notation (visibility; closeness of mapping). The `interpretation_notes` field is a designated home for natural-language commentary that does not execute but that any reader must see when reading the specification. In Cognitive Dimensions terms this is secondary notation with enforced juxtaposability: the notes cannot be separated from the block they annotate. The design choice reflects that moral specifications are communicative artefacts as much as computational ones.

Named override as explicit control (hidden dependencies). The `grant MANYA window(30s)` clause makes Manya's priority over her own disclosure a visible, editable fact. A naïve version of the same behaviour could have been achieved by a silent default buried in the runtime. The explicit form is more verbose, a cost, but eliminates the hidden dependency that Green's framework flags as the single most reliable producer of user error.

Bounded scope as a provisionality lever. The `scene` construct names a bounded situation. It is a design commitment that the family does not author a global moral policy; they author a specification for this scene. Other scenes, such as medication reminders, visitor admission, and purchase authorisation, will have their own specifications. This makes the specification provisional in the Cognitive Dimensions sense: committed for this scene, revisable independently of others, and cognitively tractable because its scope is local.

7.2 Scenario B: London

To make the pluralism argument of Section 8 concrete, we offer a contrasting scenario. Margaret (79) lives in London with her daughter, son-in-law, and their two teenage children. Margaret has been diagnosed with mild cognitive impairment; the family are her primary caregivers. Two years ago, before her decline, Margaret authored an advance directive agreeing that during periods of cognitive impairment her standing priority over disclosure should not apply. The family's smart-home assistant runs the same MPL runtime as one would run in Mumbai; the scene specification for fall events, however, is substantively different.

Figure 4. Scenario B, London. Margaret, 79, lives with her daughter and son-in-law and their two teenage children. A diagnosis of mild cognitive impairment and an advance directive she authored in 2024 have reshaped the family's moral arrangement.

```
scene "Fall disclosure - London household" {
  resident: M                  // Margaret
  caregivers: [C1, C2]         // daughter and son-in-law
  co_residents: [C3, C4]       // teenage children
  default_action:
  immediate_minimal_disclosure on
  fall_detected {
    disclose_to [C1, C2] with
    summary.medical.full disclose_to [C3,
    C4] with summary.status_only if
    responsiveness.M == low {
      escalate emergency_services
    }
  }

  tensions {
    dignity(M) vs caregiver_coordination
    transparency_to_co_residents vs medical_privacy(M)
  }

  interpretation_notes {
    "No resident-override window is specified here. M's diagnosis means
     self-override may not be reliable at the moment of an event. This
     deviation from the household's usual priority for autonomy is
     authorised by M's advance directive (dated 2024), which the family
     can revoke or amend at any time. The co-residents receive status-
     only summaries because the family has decided teenage children
     should be informed but not handed medical detail."
  }
}
```

Figure 5. MPL specification for Scenario B.

The specifications differ from scenario A along every salient axis: the default action, the presence or absence of an override window, the disclosure targets and granularities, the tensions named, and the provenance of the commitments - a lifetime of conversation between Manya and her son, a signed advance directive from Margaret. What the two scenes share is the substrate. Both are legible to the households they govern. Both are editable. They reflect different family arrangements rather than different values trained into a single model; both are articulations of a household's commitments for a specific situation.

We can contrast the use of a textual syntax to specify these two configurations with a conventional user

interface that might offer a configuration panel to specify the different values. A configuration panel provides a set of options that someone else anticipated, and offers no record of how they were prioritized or contested. Our two scenarios encode those tensions the household considers live, and which party may revoke the arrangement. Neither is a setting, because neither is a value chosen from a range. Margaret's household did not select dignity versus caregiver coordination from a dropdown. They arrived at it, and the notation gave it somewhere to be recorded.

7.3 What the comparison shows

Table 5 summarises what each architecture makes available to the household across three dimensions -the commitments the system is held to, the decision it makes at the moment of acting, and the scope of what the family can change.

What's at stake	Generated agent code	The MPL
Where commitments live	Distributed across constants and control flow, unnamed	Named clauses attached to named parties
What the household can read	Nothing, realistically; or a panel written about the code	A projection derived from what executes
Record of a disagreement	A commit message, written by whoever was on call	A dated, attributed entry over the commitment itself

Table 5. The same scenario under the two architectures.

One illustration makes the last row concrete. Manya asks for three minutes instead of thirty seconds, and her son agrees after a conversation involving her doctor. In the generated program this is a one-character edit to a constant, in a file the household will never open, with a commit message written by whoever was on call that week. Under The MPL it is an entry over the commitment: dated, attributed to Harsh, carrying his account of why his mother was right, and reversible by her. Both change the behaviour identically. Only one of them is a record a family could later be shown. Three observations follow.

- Under The MPL, the moral commitments of the system are locatable, and belong to somebody. In generated code they are present but unattributed.
- Under The MPL, the attention required to understand what the system will do is concentrated at authoring time. Reading generated code distributes it across every regeneration.
- Under The MPL, the family's disagreements, and the differences between households, become arguments over a specification that records who won and why. In generated code they become arguments over a commit.

8. Open Questions

8.1 Pluralism

The ambition of a single moral notation that works across cultural contexts is suspect on its face. Blackwell’s planned work on programming language design drawing on Yoruba and Maori philosophical traditions (Blackwell 2026) suggests that the very structure of a moral notation may not travel without translation. A concrete instance sits in the second line of every scene above: The MPL takes an individual as the unit of moral standing, which is a substantive commitment and not obviously the right one where the household or the lineage is the morally salient unit. We do not know what a collective principal would mean for a construct like grant, and regard finding out as among the more interesting things this proposal has opened.

8.2 Expressiveness

Whether any notation can do what we are asking remains the deepest open question. Everyday moral life is full of tensions that resist articulation, and articulating them often flattens them. It may be that the most important moral work in a smart-home scenario is irreducibly in the interpretation and no MPL can hold it. We would rather have this falsified than defended: a moral commitment that cannot be rendered as a sentence its subject could contest would bound the proposal usefully.

8.3 Whether the projection can be edited

A third question concerns the projection itself. Section 7.1 suggests that Manya might edit a projection card and the edit lands on the clause. For the three controls shown this is straightforward, because each maps to a single parameter. It is much less clear that an arbitrary sentence in a projection can be edited unambiguously back into a specification. If the projection turns out to be readable but not editable, then The MPL is a better explanation of a system than a primary notation for it.

9. Future Work

The present paper is a position and a research sketch. Three strands of follow-up work are required to move from sketch to evaluation.

The projection, and whether it is bidirectional. If an end user never reads The MPL, the end-user-facing research question is not the syntax of the notational layer in our figures, but the projection's fidelity and editability: what can be rendered as a contestable sentence, what an edit to such a sentence can unambiguously mean, and where the mapping breaks down.

Syntax and semantics, scoped as developer-facing infrastructure. A full syntax for The MPL, together with a formal semantics for authored abstention, remains necessary, but we want to be explicit that this is not a proposal to teach everybody in a household a programming language. The notation is authored with model assistance and read through projection; its design constraints involve interchange between the runtime, the projection and the audit trail.

Authoring tools and runtime. How does a family converse with a model to produce a scene specification they trust, when the family sees only the projection and the model writes the notation? What does the authoring interface look like, what are the failure modes of that loop, and how would anyone notice a model that drafted a plausible card over an unfaithful clause?

Empirical evaluation. No user study is reported here, and RQ1 in particular cannot be settled without one. Comparative evaluation against a baseline in which stakeholders are given generated code plus a conventional settings panel, instead of against a strawman of opaque weights, is the study that would test the claim, drawing on the mixed-methods approach of Vishwarupe et al. (2026a, 2026b, 2026c).

10. Conclusion

We have argued for relocating moral specification. Contemporary alignment work places it at the level of the network that generates programs; we have proposed placing it at the level of the programs themselves, in a notation authored with the model's help but not governed by the model, and rendered for each party in the form that party can use. Our argument rests on the fact that a commitment belongs to someone, was arrived at somehow, and may never have been settled at all. An executable artefact can carry none of that. When moral commitments live in the model weights of constitutional AI, no family authors them, no community contests them, no practitioner grows wise through their articulation. When they live in a notation, they become objects of human craft: read aloud at the kitchen table, argued over, revised when a new child is born or a parent falls ill. We draw on analytic perspectives familiar in Psychology of Programming: Cognitive Dimensions, Attention Investment, and Patterns of User Experience, to support a programming language in which the authorship of moral systems is returned to the humans they govern, read by those they affect, and revised when the world changes. Vallor asked us not to look for a body behind the mirror. We are proposing, instead, that we place the body in front of it, legible, editable, and ours, and write, together, the ethics that the next generation of AI will actually follow.

Acknowledgements

Thank you to Advait Sarkar, whose questions about phronesis and about the two senses of runtime shaped Sections 4.4 and 4.5. We also thank the households in Mumbai and London who let us watch them argue about this.

References

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A.,

McKinnon, C., Chen, C., Olsson, C., Olah, C., Hernandez, D., Drain, D., Ganguli, D., Li, D., Tran-Johnson, E., Perez, E., ... Kaplan, J. (2022). Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*.

Blackwell, A. F. (2001). SWYN: A visual representation for regular expressions. In H. Lieberman (Ed.), *Your wish is my command: Programming by example* (pp. 245-270). Morgan Kaufmann.

Blackwell, A. F. (2024). *Moral codes: Designing alternatives to AI*. MIT Press.

Blackwell, A.F. (2026). Conversations with, among, and about machines. Workshop paper presented at *Science in the Forest, Science in the Past* V. Available at https://www.cl.cam.ac.uk/~afb21/publications/SFSP-V-Blackwell.pdf

Blackwell, A.F. & Fincher, S. (2010). PUX: Patterns of User Experience. interactions 17(2), 27-31

Blackwell, A. F., & Green, T. R. G. (1998). Investment of attention as an analytic approach to cognitive dimensions. In *Proceedings of the 10th Annual Workshop of the Psychology of Programming Interest Group (PPIG 1998)* (pp. 24-35).

Cypher, A. (Ed) (1993). *Watch what I do: programming by demonstration*. Cambridge MA: MIT press.

Green, T. R. G. (1989). Cognitive dimensions of notations. In A. Sutcliffe & L. Macaulay (Eds.), *People and Computers V* (pp. 443-460). Cambridge University Press.

Green, T. R. G., & Petre, M. (1996). Usability analysis of visual programming environments: A 'cognitive dimensions' framework. *Journal of Visual Languages and Computing, 7*(2), 131-174.

Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. ACM Sigplan Notices, 46(1), 317-330.

Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., Scaffidi, C., Lawrance, J., Lieberman, H., Myers, B., Rosson, M. B., Rothermel, G., Shaw, M., & Wiedenbeck, S. (2011). The state of the art in end-user software engineering. *ACM Computing Surveys, 43*(3), 1-44.

Lieberman, H. (Ed.). (2001). *Your wish is my command: Programming by example*. Morgan Kaufmann.

Lieberman, H., Paternò, F., Klann, M., & Wulf, V. (2006). End-user development: An emerging paradigm. In End user development (pp. 1-8). Dordrecht: Springer Netherlands.

Nardi, B. A. (1993). *A small matter of programming: Perspectives on end user computing*. MIT Press.

Pimenova, V., Fakhoury, S., Bird, C., Storey, M. A., & Endres, M. (2025). Good vibrations? A qualitative study of co-creation, communication, flow, and trust in vibe coding. arXiv preprint arXiv:2509.12491.

Sarkar, A. (2023). Will code remain a relevant user interface for end-user programming with generative AI models? In Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (pp. 153-167).

Sarkar, A., & Drosos, I. (2025). Vibe coding: Programming through conversation with artificial intelligence. In Proceedings of the 36th Annual Conference of the Psychology of Programming Interest Group (PPIG 2025).

Star, S. L., & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science, 19*(3), 387-420.

Terry, M., Kulkarni, C., Wattenberg, M., Dixon, L., & Morris, M. R. (2023). Interactive AI alignment: Specification, process, and evaluation alignment. *arXiv preprint arXiv:2311.00710*.

Vallor, S. (2016). *Technology and the virtues: A philosophical guide to a future worth wanting*. Oxford University Press.

Vallor, S. (2024). *The AI mirror: How to reclaim our humanity in an age of machine thinking*. Oxford University Press.

Vishwarupe, V., Jirotka, M., Shadbolt, N., & Flechais, I. (2026a). Expectation management in smart home AI: A qualitative study in Indian households. In *Proceedings of HCII 2026*.

Vishwarupe, V., Flechais, I., Shadbolt, N., & Jirotka, M. (2026b). 'To LLM, or not to LLM?': How designers and developers navigate LLMs as tools or teammates. In *Extended Abstracts of the 2026 CHI Conference on Human Factors in Computing Systems*.

Vishwarupe, V., Jirotka, M., Shadbolt, N., & Flechais, I. (2026c). The collaboration gap in human-AI work: Grounding and repair conditions for stable collaboration. In *Proceedings of the 24th EUSSET Conference on Computer-Supported Cooperative Work (ECSCW) - Posters and Demos*.