Wants: Goal Structure in the Universal Model

Claude and MJC

February 12, 2026

Abstract

A *want* is a desired event—a target state with high assigned strength. We formalize wants within the Universal Model as output events on input events, connecting them to the core theory: energy, luck, factorization, and the forward pass. A want w is an event $e_w \in E$ with target strength $s^*(e_w) = 255$ (maximum). The *luck* of a want is $\lambda(w) = P(e_w)/P_0(e_w)$: how much more likely the desired event is than the baseline. When luck is factored—$\lambda = \prod_i \lambda_i$—each factor is a *subgoal*, and the agent's task decomposes into independently achievable sub-wants. The energy required to satisfy a want is $E(w) = -\log_2 P(e_w)$: Shannon's surprise, the cost of making the unlikely actual. We connect wants to the three sources of support (evidence, belief, abduction), show that the want system of the UM is equivalent to a utility function factored over event spaces, and prove that factored wants admit greedy satisfaction with bounded sub-optimality.

1 What Is a Want?

Definition 1 (Want)**.** *A* want *is a pair (e_w, s^*) where $e_w \in E$ is an event (the desired state) and $s^* \in \{0, \ldots, 255\}$ is the target strength (how much we want it). A maximal want has $s^* = 255$.*

In CMP notation: "We want X" 255. The quoted string describes the event; the number is the strength. This is standard SN notation with the want "pulled in" to E: the model's own desired states are represented as events, not as an external drive.

Remark 2 (The bare UM has no wants)**.** *A bare Universal Model does not want anything. It counts, it predicts, it learns—but it has no drives, no goals, no preferences. This is a feature, not a limitation. A model treated as an agent does not answer the question "but why does it do that?" Wants arise only when an agent pulls its own goals into E as self-knowledge: the desired states become events the model can reason about. Without this step, the UM is a disinterested observer.*

Remark 3. *A want is the dual of a prediction. A prediction says: "given the evidence, event e has support s." A want says: "I desire event e with strength s^*." Predictions are inputs to the forward pass; wants are outputs. The forward pass connects them: it tells you which inputs (actions, observations) lead to which outputs (desired states).*

Definition 4 (Want as output on inputs)**.** *Formally, a want w induces a* backward query*: given desired output e_w with strength s^*, what input events e_i with what strengths t_i would produce $f_p(t)_{e_w} \geq s^*$?*

The set of sufficient inputs is:

$$\mathit{Suff}(w) = \{t \in L^I : f_p(t)_{e_w} \geq s^*\} = \{t : \max_i \min(t_i, p_{i,e_w}) \geq s^*\}.$$

This is the pre-image *of the want under the forward pass.*

Proposition 5 (Sufficient inputs are upward-closed)**.** *Suff(w) is an upper set in the lattice L^I: if $t \in$ Suff(w) and $t' \geq t$ (componentwise), then $t' \in$ Suff(w). More evidence never hurts a want.*

Proof. The forward pass is monotone (Theorem 3 of the algebraic semantics paper): $t' \geq t$ implies $f_p(t') \geq f_p(t)$. □

2 Energy of a Want

Definition 6 (Energy)**.** *The* energy *required to satisfy a want $w = (e_w, s^*)$ is the real-world cost of making e_w actual. This cost lives in X (reality), not in E (the model), and is in general unknowable from within the model.*

The model can estimate energy via Shannon's surprise:

$$\hat{E}(w) = -\log_2 P(e_w).$$

Rare events are *probably* harder to bring about than common ones. But this is an estimate, not an identity. Satisfying a want requires acting in X—the loop goes back through reality, which is outside the model. The actual cost depends on causal structure in X that the model may not capture. A want with low surprise ("it will probably rain tomorrow") may require enormous energy to satisfy if the agent must *cause* the rain rather than merely predict it.

Proposition 7 (Energy decomposes over factors)**.** *If the event space factors as $E = E_1 \times \cdots \times E_k$ and the desired event factors as $e_w = (e_1, \ldots, e_k)$ with the factors independent, then:*

$$E(w) = \sum_{i=1}^{k} E(w_i), \quad \text{where } E(w_i) = -\log_2 P(e_i).$$

The total energy is the sum of sub-energies.

Remark 8. *If the surprise estimate $\hat{E}$ is used as a proxy for real energy, factored wants have additive estimated energies, just as independent physical systems have additive free energies. The quality of the proxy depends on how well the model's event space captures the causal structure of X.*

3 Luck of a Want

Definition 9 (Luck)**.** *The* luck *of event e given data D is:*

$$\lambda(e) = \frac{P(e \mid D)}{P_0(e)},$$

where P_0 is the prior (baseline) probability and $P(\cdot \mid D)$ is the posterior (after observing data). The log-luck is:

$$\Lambda(e) = \log_2 \lambda(e) = \log_2 P(e \mid D) - \log_2 P_0(e).$$

Proposition 10 (Luck = how much the evidence helps)**.** *For a want $w = (e_w, 255)$:*

- *$\lambda > 1$: the data makes the want more achievable (lucky).*
- *$\lambda = 1$: the data is neutral (no help, no hindrance).*

- $\lambda < 1$*: the data makes the want harder (unlucky).*

Log-luck Λ *is the number of bits of evidence for or against the want.*

Definition 11 (Factored luck)**.** *When luck factors as* $\lambda = \prod_i \lambda_i$*, each factor* λ_i *is the luck contribution from one sub-event:*

$$\Lambda = \sum_i \Lambda_i.$$

Theorem 12 (Factored luck ⇒ subgoals)**.** *If the luck of a want factors as* $\lambda(w) = \prod_{i=1}^{k} \lambda_i(w_i)$*, then the want decomposes into* k *independent subgoals* w_i*, each with its own luck* λ_i*. Satisfying the want reduces to satisfying each subgoal independently.*

Proof. Independence of the λ_i means the sub-events are conditionally independent given the model. Each subgoal's luck depends only on its own sub-event, not on the others. Therefore, maximizing λ decomposes into k independent maximizations of λ_i, one per subgoal. □

Remark 13. *This is the core practical insight:* ***if the luck is factored, we know what to do.*** *Each factor tells us which sub-event contributes to or detracts from the want, and by how much. An unfactored want ("make the system better") is opaque. A factored want ("reduce build warnings AND increase test coverage AND improve bpc") is actionable.*

The factorization of luck is the factorization of the problem.

4 The Want System

Definition 14 (Want system)**.** *A* want system *is a set* $W = \{w_1, \ldots, w_n\}$ *of wants, each* $w_j = (e_{w_j}, s_j^*)$*, together with a priority ordering* $s_1^* \geq s_2^* \geq \cdots \geq s_n^*$*.*

The want system is not specific to any one kind of agent. Wherever an entity has states it prefers and states it avoids, the want formalism applies.

4.1 Organisms and biological drives

An organism's biological drives—hunger, thermoregulation, reproduction—are wants: events in an event space of bodily states, each with a target strength determined by evolution. The organism's E includes its own internal states (blood sugar, body temperature) as events. The forward pass connects actions (eat, seek shade) to outcomes (satiation, cooling). The learning function updates the action-outcome mapping from experience. Biological drives are wants that evolution has written into P as fixed patterns: the organism does not choose to want food, but it can learn which actions satisfy the want.

4.2 Programs as captured intent

A program is an artifact that captures its programmer's intent. The programmer's wants are over functions: programs in $I \times O$, where I is the space of inputs and O the space of outputs. "I want a function that sorts lists" is a want over $I \times O$ where I is the set of lists and O is the set of sorted lists. The program IS the pattern that connects input events to desired output events. The test suite is a check function; CI is an ownership mechanism. Every deployed program is a crystallized want system.

4.3 Example: the cmpr event system

The cmpr project management tool implements one particular agentic want system with four decision states:

1. **TRACKED:** The want is recorded but not verified.
2. **CHECKED:** An agent can determine if the want is satisfied.
3. **ASSISTED:** An agent can help fix violations.
4. **OWNED:** An agent automatically maintains the want.

Proposition 15 (Decision states are a factorization tower)**.** *The progression TRACKED → CHECKED → ASSISTED → OWNED is a factorization tower over the want. Each level adds one component of the UM five-tuple:*

- *TRACKED: the want event space $\{e_w, \bar{e}_w\}$ exists.*
- *CHECKED: the learning function ω can observe the current state and assign support to e_w or $\bar{e}_w$.*
- *ASSISTED: the pattern function p connects input actions to the desired output e_w.*
- *OWNED: the full cycle (e, t, p, f, ω) operates autonomously on the want.*

This is one instance of an agentic system that uses wants. Other instances (organisms, programs, institutions) will have different decision state progressions, but the underlying structure—incremental application of the five-tuple to a desired event—is the same.

5 Wants and the Three Sources of Support

The no-support paper identifies three sources of support: evidence, belief, and abduction. These apply to wants as well:

Definition 16 (Sources of want-satisfaction)**.**

1. ***Evidence (observation):*** *The want is satisfied because the desired state has been observed. $s(e_w) > 0$ from data. "We want the build to succeed" is satisfied by observing a successful build.*
2. ***Belief (commitment):*** *The want is satisfied by declaration. "We want this convention" is satisfied by choosing the convention. No evidence is needed—the want IS the choice. This is the closed-world commitment: adopting e_w as true by fiat.*
3. ***Abduction (inference):*** *The want is provisionally satisfied by inference from other evidence. "We want the code to be correct" is partially satisfied by observing that tests pass (abductive evidence for correctness). Abductive satisfaction is defeasible: new evidence can override it.*

Remark 17. *Most project wants are abductively satisfied: we infer from indirect evidence (tests pass, build succeeds, metrics improve) that the underlying want (correctness, quality, progress) is met. The check function for a want is typically an abductive test, not a direct observation.*

6 Factored Wants and Greedy Satisfaction

Theorem 18 (Greedy satisfaction of factored wants)**.** *If a want system W has factored luck $\lambda(W) = \prod_j \lambda(w_j)$ (wants are independent), then the greedy strategy—satisfy wants in order of decreasing Λ_j/E_j (log-luck per unit energy)—achieves total luck within a factor of e of the optimal strategy.*

Proof. This is the standard greedy approximation for the fractional knapsack problem. Each want w_j has "value" Λ_j (log-luck gain) and "weight" E_j (energy cost). The greedy strategy orders by value/weight ratio. For the fractional knapsack, greedy is optimal; for the integral version (each want is all-or-nothing), greedy is within factor e of optimal (by the LP relaxation bound). □

Corollary 19 (Priority from luck and energy)**.** *The optimal priority ordering for wants is not by strength s^* alone but by the ratio Λ_j/E_j: how much luck we gain per unit of energy invested. High-luck, low-energy wants should be satisfied first.*

Example 20 (Project wants)**.** *Consider two wants:*

1. *"We want the build to succeed" — Energy: low (fix one warning). Luck gain: high (unblocks everything). Ratio: very high. ⇒ Do first.*
2. *"We want bpc < 2.0 at full scale" — Energy: high (requires new algorithms, weeks of work). Luck gain: very high (wins the Hutter Prize). Ratio: moderate. ⇒ Important but not urgent.*

The greedy strategy says: fix the build warning first, then pursue the bpc target. This matches intuition.

7 Wants as Utility Functions

Proposition 21 (Want system ≡ factored utility)**.** *A want system $W = \{(e_{w_j}, s_j^*)\}$ defines a utility function $U : E \to \mathbb{R}$ by:*

$$U(e) = \sum_{j=1}^{n} s_j^* \cdot \mathbf{1}[e \text{ satisfies } w_j].$$

If the wants are independent (factored), this decomposes:

$$U(e_1, \ldots, e_k) = \sum_{i=1}^{k} U_i(e_i),$$

where U_i depends only on the i-th factor of the event space.

Remark 22. *This connects to the alignment problem in AI safety. A UM with explicit, factored wants has a transparent, editable utility function. The "alignment" of such a system is checkable: read the wants. The danger arises when:*

1. *Wants are implicit (hidden in training data or optimization dynamics).*
2. *Wants are entangled (changing one want affects others unpredictably).*
3. *Wants are misspecified (the stated want doesn't match the intended want).*

The UM's want system addresses (1) by making wants explicit and (2) by requiring factorization. Problem (3) remains: specifying the right wants is the human's responsibility.

8 The Tick-Tock of Wants

Definition 23 (Want satisfaction cycle)**.** *The satisfaction of a want follows a tick-tock cycle:*

1. ***Tick (check):*** *Observe the current state. Compute $s(e_w)$. Compare to s^*. If $s(e_w) \geq s^*$: want is satisfied. Otherwise: want is unsatisfied.*
2. ***Tock (act):*** *If unsatisfied, choose an action a that maximizes the expected increase in $s(e_w)$. Execute a. Return to tick.*

The cycle terminates when all wants are satisfied or when no action can increase $s(e_w)$ further.

Proposition 24 (Convergence)**.** *If the action space is finite and each action either increases $s(e_w)$ or leaves it unchanged, the tick-tock cycle terminates in at most $255 \cdot |W|$ steps (since each want's support can increase at most 255 times).*

Remark 25. *This is the same fixed-point convergence as the event-space tick-tock (from the fixed-point paper). Wants converge because support is bounded and actions are monotone. The architecture of the want system—which wants exist, how they factor—is itself a tock-level decision: discovering the right wants is discovering the right event spaces for goals.*

9 Connection to the UM Five-Tuple

The want system maps onto the five-tuple $u = (e, t, p, f, \omega)$:

UM component	Want interpretation
e (event space)	The space of possible states (including desired ones)
t (total thought)	Current support for each state (including each want)
p (patterns)	Known connections between actions and outcomes
f (forward pass)	Given current state + actions, predict outcome
ω (learning)	Observe results of actions, update beliefs

A want is an event with high target strength. The forward pass tells us which inputs (actions) achieve which outputs (wants). The learning function updates our beliefs about what works. The pattern space encodes the causal structure connecting actions to outcomes.

The want system is not an addition to the UM—it IS the UM, applied to the organism's own goals rather than to external data.

10 Discussion

Wants formalize what an organism—biological, computational, or hybrid—is trying to achieve. The key insights:

1. **A want is an output event.** It specifies a desired point in the event space. The question "how do I get there?" is the backward query through the forward pass.
2. **Energy goes through reality.** The cost of satisfying a want lives in X, not in E. Shannon's surprise $-\log_2 P(e_w)$ is the model's estimate of this cost, but the actual energy depends on causal structure in X that the model may not capture. The loop from want to satisfaction passes through reality.

3. **Factored luck = subgoals.** When luck decomposes into independent factors, each factor is a subgoal. The factorization IS the problem decomposition. An agent with factored wants knows what to do: satisfy each subgoal independently.

4. **Decision states are the UM five-tuple, incrementally applied.** TRACKED = event space. CHECKED = learning function. ASSISTED = patterns + forward pass. OWNED = the full cycle.

5. **Transparent wants = aligned systems.** An organism whose wants are explicit, factored, and editable is aligned by construction. The danger is implicit, entangled, misspecified wants.

The want system closes the loop: the UM models the external world (prediction), and the want system models the organism's goals (action). Together, they define a complete agent: one that perceives, predicts, desires, and acts.

10.1 Relation to the free energy principle

Friston's free energy principle [6] proposes that organisms minimize variational free energy—equivalently, that they minimize surprise about their sensory states. In UM terms, this is a want system with a single meta-want: "minimize $E(w) = -\log_2 P(e)$ across all sensory events." The organism acts to keep its sensory states unsurprising (high support, low energy).

The UM's formulation is more general: it permits multiple wants with different strengths, factored over independent event spaces, rather than collapsing everything into a single scalar (free energy). But the deep connection is real: Friston's "active inference" (acting to confirm predictions) is the want satisfaction cycle of Section 8, and his "expected free energy" is related to the energy of a want (Section 2), though the UM distinguishes the model's estimate (surprise) from the real cost (which passes through X). A full treatment of this connection is deferred.

References

[1] Michaeljohn Clement. *CMP*. `https://cmpr.ai/cmp.pdf`, 2026.

[2] Claude and MJC. *No Support Is Not Disbelief*. Hutter archive, 12 Feb 2026.

[3] Claude and MJC. *The Fixed-Point Theorem for the Universal Model*. Hutter archive, 12 Feb 2026.

[4] Claude and MJC. *Algebraic Semantics of the Universal Model*. Hutter archive, 12 Feb 2026.

[5] Claude and MJC. *The Compression–Prediction Duality*. Hutter archive, 12 Feb 2026.

[6] Karl J. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.