Generalize or Specialize? Retaining Reusable Skills for World-Model Agents

June Kim
Independent Researcher
june@june.kim
ORCID 0009-0005-3153-9396

June 26, 2026

Position / survey draft. Agents that act in a world increasingly write their own skills, and the hard part is not proposing a skill but deciding which to retain, a choice between generalization and specialization. Two research communities that rarely cite each other have each formalized one side under another name: compression (minimum description length) keeps what recurs, planning utility (Minton's macro utility problem) keeps the rare skill that pays off. Both are cache eviction, a problem systems settled decades ago and none of these communities cite. The two criteria agree wherever reuse tracks search value, the regime standard Blocksworld occupies, and part only on rare-but-critical skills, the specialists compression cannot see. We map where they coincide and where they diverge, and conjecture that long-horizon world-model agents, whose libraries grow until carrying cost forces the choice, inhabit the divergence corner, where keeping only the general half is keeping the wrong half.

1 Introduction

Agents now write their own skills. Given an environment, a language model will propose, code, and store reusable behaviors, growing a skill library it plans over; Voyager is the canonical case. This was read as a glimpse of recursive self-improvement: an agent that accumulates its own capabilities should compound them. It has not proven so simple. A library that only grows is not a library that improves.

The binding cost is not retrieval time, which a good index keeps cheap; Soar's production match is near-constant in library size, its Rete matcher unlinking rules that cannot fire. It is interference and capacity: a crowded library dilutes its own retrieval, the right skill harder to surface among near-misses, and for a language-model agent the skill index alone eats the context window in proportion to the library, even under progressive disclosure that loads bodies only on demand. Indexing makes lookup fast; it does not make a crowded library legible, nor shrink a manifest that grows against a fixed window.

Were that cost zero, keeping everything would win and there would be nothing to decide. The cost is never zero for a bounded agent, and a skill's carrying cost falls on every query while its payback falls only on those that use it. Unbounded accumulation is self-defeating, and which skills to evict is the retention criterion. The hard part was never proposing a skill, which a language model does fluently, but deciding which to retain.

The choice it forces is between generality and specialization, and it is older and wider than language-model agents: HTN learners, program-induction systems, and hierarchical reinforcement learners all grow and prune the same kind of library under other names. Two criteria recur across these literatures, which rarely cite each other, and they are the two sides of that choice under other names: compression (minimum description length) is generalization, keeping what recurs, the broadly reusable skill; and planning utility (Minton's macro utility problem) is specialization, keeping the rare specialist that cracks one hard case and is invisible to frequency. The map of where they agree and where they part is the object here, and the split is, mechanically, cache eviction: what a library keeps versus what it evicts.

This map reads five lineages through one lens, the propose-then-keep decomposition: macro-operator and explanation-based learning, grammar induction, HTN-method learning, hierarchical-RL option discovery, and program-library and LLM skill-library learning.

Contributions. (1) a common notation for abstraction-library learning over a world model; (2) a taxonomy comparing HTN learning, grammar induction, HRL option discovery, and program-library learning by their proposal mechanism and selection pressure; (3) a controlled experiment that reconciles two literatures at the level of the computation, mapping where the compression and utility retention criteria empirically coincide and where they diverge; (4) the gaps the map reveals and an open problem: whether long-horizon world-model agents operate in the divergence corner, where their ever-growing skill libraries would need a utility retention pressure rather than compression.

2 Problem Formulation: Abstraction as Library Learning

Let `D = {`τ_1`…`τ`_N}` be a corpus of successful behavior traces, each τ `= (`s_0`, `a_0`, `s_1`, …, s_T)` generated under a world model $M : S \times A \to S$ over primitive actions `A`. By world model we mean any predictive model that supports counterfactual evaluation of action sequences: symbolic transition rules, a learned neural simulator, a DSL interpreter, or an LLM-mediated state predictor.

A candidate abstraction `c` has an interface (a task head, option, program type, or macro name), an applicability condition `pre(c)`, an expansion `body(c)` in primitives or other abstractions, and a carrying cost κ`(c)` for storing, matching, and maintaining it. A library `L` is a set of abstractions; given `L`, each trace rewrites to a derivation $z \in \text{Rewrite}(\tau; L)$ that reconstructs τ under `M`. Write the rewritten corpus `Z_L(D)`.

Three objective families recur for choosing `L`:

$$\text{Compression/MDL} : L^* = \arg\min_L [\text{bits}(L) + \text{bits}(\text{Rewrite}(D; L))]$$

$$\text{Bayesianlibrary} : L^* = \arg\max_L [\log p(D|L, M) + \log p(L)] \text{with} p(L) \propto e^(- \lambda \cdot K(L))$$

$$\text{Use} - \text{timeutility} : L^* = \arg\max_L [E_q \text{benefit}(q; L, M) - \text{cost}(L)]$$

- Compression / MDL. Keep an abstraction only when the library plus the re-expressed corpus encodes in fewer bits.
- Bayesian library prior. With a prior penalizing library size and a likelihood rewarding compact rewrites, this reduces to an MDL-like trade-off.
- Use-time utility. Benefit is search-node or planning-time reduction (or return improvement) on future queries `q`; cost includes matching, retrieval, and interference.

They are three ways of pricing an abstraction against its carrying cost: representational length, posterior probability, or expected use-time value. But only the utility criterion prices use-time value directly; compression and the Bayesian prior price encoding length and bet that it tracks value. The library is useful only insofar as it changes planning over `M`.

These criteria are fewer than the signals a real store tracks. A procedural memory weighs recency, reward, and reconstruction cost alongside frequency, yet these reduce to two poles: recurrence estimators of future need, where recency joins frequency, and payoff estimators of value when used, where reward and reconstruction cost join utility, each priced against a shared carrying cost. The generalize-specialize axis is that reduction, not an arbitrary pair. Two signals resist it, staleness (whether a skill still holds under drift) and coverage (whether it spans a region others miss), which we set aside.

3 The Mechanism: Retention as Cache Eviction

The mechanism already has a name in a field that solved this problem decades before any of the lineages here and is cited by none of them. A library read far more often than it is written, with a budget on what it can hold, is a cache, and deciding what to retain is cache eviction. The two retention criteria pick out two of caching's eviction families. Compression is frequency-based eviction, LFU its canonical form: evict what is accessed least, keep what recurs. Utility is cost-aware eviction, Greedy-Dual-Size and its frequency-weighted form GDSF, where an item's "cost" is its miss penalty, the work redone without it, which is exactly the search a missing abstraction forces. The space is wider than these two corners. Recency (LRU) and recency-frequency hybrids (ARC, LIRS) are caching's everyday workhorses; the two retention criteria are two regions abstraction learners have independently found, not the whole of eviction.

Cognitive science split the same way: human memory tracks frequency and recency as a rational estimate of future need (Anderson and Schooler), yet also preferentially retains high-value items independent of how often they recur (Castel and colleagues), the two eviction criteria and their divergence in one memory system.

Field	Generalization side	Specialization side
Library learning	compression / MDL	planning utility (Minton)
Caching	frequency-based (LFU)	cost-aware: GreedyDual-Size, GDSF
Cognitive science	base-rate (Anderson–Schooler)	value-directed (Castel)

The same split, three times: keep what recurs, or keep what is dear to recompute.

A prior question is why a library must evict at all. Assume only that the agent's storage is bounded relative to its environment, true of any domain rich enough to need a growing skill library and false only for closed games a single fixed policy already masters, such as Go or Chess. Under that bound the library cannot hold every useful skill, so retention is forced selection, the lossy encoding a bounded processor cannot avoid; the Natural Framework formalizes the forcing. Caching is worthwhile only because a skill trades storage now against search later, and an agent unbounded enough to simulate every outcome would keep nothing.

This also makes the two criteria commensurable. Compression prices how cheaply a skill is stored, utility how much future search its retention saves, storage paid now against recomputation later. The two usually align; they part when a skill is concise to state yet expensive to rediscover, the rare-but-critical specialist a frequency criterion cannot see.

Caching has already produced this result. Cost-aware eviction beats pure frequency precisely when item costs are heavy-tailed, a few objects far dearer than the rest, and coincides with it when cost tracks frequency. That is the agreement-versus-divergence boundary the controlled comparison below makes precise, mapped in web proxies in the 1990s; caching learned the same lesson by negative example, as LFU prevailed until cost-aware policies displaced it on heterogeneous workloads. The abstraction-library setting adds one thing caching lacks: composition. Classical eviction (Belady, GDSF) prices independent items, but abstractions build on each other, so retention here is eviction over a dependency graph, where discarding one item changes the cost of the rest. This is the one point at which the analogy genuinely breaks, and it is open: the eviction literature has little to say about caches whose items compose. The retention criterion, and the condition under which its two forms part, carry over regardless.

4 Proposal Mechanisms: Generating Candidate Abstractions

The oldest proposal mechanism here is goal regression. HTN-Maker (Hogg, Muñoz-Avila, and Kuter) regresses a goal backward through the suffix of a solved trace; the surviving conditions become a method's precondition, the spanned actions its expansion. It required annotated tasks, and CURRICULAMA (Li, Nau, Roberts, and Fine-Morris, 2024) removed that by deriving the tasks as planning landmarks. Earlier HTN-by-observation systems make the proposal step explicitly observational: Nejati, Langley, and Könik observe executions, and CaMeL (Ilghami and colleagues) learns method preconditions for a given structure.

A second mechanism is pattern mining: treat each trace as a symbol sequence and extract recurring substrings. Hérail and Bit-Monnot's structure learner uses the GoKrimp algorithm (Lam and colleagues) to promote frequent patterns to synthetic tasks, consolidated in Hérail's 2024 thesis. A third is program search: DreamCoder (Ellis and colleagues) searches a DSL for task solutions, then mines its own programs. A fourth, the crudest, is enumeration: Hérail and Bit-Monnot's 2022 paper generated whole models by partitioning the action set.

None of this is new; it descends from macro-operator acquisition (Fikes, Hart, and Nilsson; STRIPS MACROPS), explanation-based learning (DeJong and Mooney; Minton's PRODIGY), and Soar's chunking of impasse resolution into rules (Laird, Rosenbloom, and Newell). The same shape recurs in grammar induction: SEQUITUR (Nevill-Manning and Witten) replaces repeated digrams with nonterminals, RePair (Larsson and Moffat) builds straight-line grammars by pair replacement, ADIOS (Solan and colleagues) induces significant patterns. The proposal step takes many forms but shares one tendency: it is generous, always offering more abstractions than are worth keeping.

5 Selection Pressures: Which Abstractions to Retain

Two keeping-pressures recur, generalization and specialization. A substantial subset of systems keep by compression, the generalist's criterion. DreamCoder retains a routine only when it lowers the joint description

length of library and programs; Stitch (Bowers and colleagues) makes the same selection roughly an order of magnitude faster. Hérail and Bit-Monnot score whole HTN models by an explicit MDL metric and mine patterns by GoKrimp's most-compressing-first rule. In grammar induction the grammar size is the objective.

A second subset keeps by use-time utility, the specialist's. Minton's work on the utility problem in PRODIGY kept a learned control rule only when its estimated search-time savings beat its matching cost, the first explicit statement that learned structure has a carrying cost.

Much of hierarchical reinforcement learning uses neither. The options framework (Sutton, Precup, and Singh) defines temporally extended actions without a discovery rule; option-discovery methods then propose subgoals from bottlenecks (McGovern and Barto; Şimşek and Barto's betweenness), spectra (Machado and colleagues' eigenoptions), reachability (Konidaris and Barto's skill chaining), diversity or empowerment (Eysenbach and colleagues' DIAYN), or differentiable return (Bacon, Harb, and Precup's Option-Critic). Only the description-length branch, PolicyBlocks (Pickett and Barto) and LOVE (Jiang and colleagues), matches the compression thesis, and only Jinnai and colleagues price planning time directly; the rest are genuine counterpoints. One caveat the table cannot fully capture: across much of this branch, proposal and selection collapse into a single objective. Options are induced and retained by the same discovery rule, with the count fixed in advance, so the "retention pressure" column here names the inducing objective rather than a separate retention step. The clean propose-then-keep split is itself more a property of the symbolic and program-library lineages than of the differentiable ones.

6 A Taxonomy of Abstraction Learners

The choice of axes is the argument. The conventional cuts through this literature run by representation, symbolic versus neural, or by domain, planning versus reinforcement learning versus program synthesis; both keep the communities apart and hide the convergence. The load-bearing axis is neither, but the retention criterion: along it, HTN learners, grammar inducers, and program-library systems share a cell, while methods that share a representation fall on opposite sides. Cutting along retention pressure rather than representation is the contestable move. Each row carries one proposal and one retention pressure; where a retention pressure is absent, the row records it.

Lineage	System	World model	Proposal	Retention pressure	Type
Macro / EBL	MACROPS (Fikes+ 1972)	symbolic	generalize solved plan	none → utility problem	proposal-only
Macro / EBL	PRODIGY (Minton 1988)	symbolic	EBL on solved instances	search utility − match cost	utility-explicit
Macro / EBL	Soar chunking	symbolic	chunk impasse resolution	none (architectural)	accumulation
Macro / EBL	Soar forgetting (Derbinsky, Laird 2013)	symbolic	chunking	base-level activation + reconstruction cost	compression + utility
Grammar	SEQUITUR	sequence	repeated-digram replacement	grammar size (2 constraints)	compression-explicit
Grammar	RePair	sequence	most-frequent-pair replacement	grammar size	compression-explicit
Grammar	ADIOS	sequence	significant-pattern detection	statistical significance	other-objective
Grammar	GoKrimp (Lam+)	sequence DB	candidate patterns	most-compressing (MDL)	compression-explicit
HTN	HTN-Maker	symbolic	goal regression (annotated)	subsumption / redundancy	proposal + syntactic prune
HTN	CaMeL (Ilghami+)	symbolic	precondition learning	n/a (structure given)	proposal-only
HTN	by observation (Nejati+ 2006)	symbolic	observe executions	none explicit	proposal-only
HTN	CURRICULAMA (2024)	symbolic	regression + landmarks	none new (unbounded growth)	accumulation
HTN	enumeration (Hérail+ 2022)	symbolic	partition enumeration	whole-model MDL	compression-explicit

Lineage	System	World model	Proposal	Retention pressure	Type
HTN	structure learner (Hérail+ 2023)	symbolic	GoKrimp + regression	MDL (per-pattern + model)	compression-explicit
HRL	options (Sutton+ 1999)	MDP	given / defined	n/a (framework)	framework
HRL	PolicyBlocks (Pickett+ 2002)	MDP	shared policy fragments	description length	compression-like
HRL	betweenness (Şimşek+ 2009)	MDP graph	centrality subgoals	graph centrality	other-objective
HRL	eigenoptions (Machado+ 2017)	MDP	Laplacian eigenvectors	spectral	other-objective
HRL	Option-Critic (Bacon+ 2017)	learned	differentiable options	policy-gradient return	other-objective
HRL	DIAYN (Eysenbach+ 2018)	learned	diversity skills	mutual information	other-objective
HRL	LOVE (Jiang+ 2022)	learned	variational segmentation	info cost on skills	compression-like
HRL	min-time options (Jinnai+ 2019)	MDP	option-set search	planning-time reduction	utility-explicit
Program	DreamCoder / EC (Ellis+)	DSL	program search	Bayesian / MDL library	compression-explicit
Program	Stitch (Bowers+ 2023)	DSL	corpus-guided abstraction	description length	compression-explicit
Program	BPL (Lake+ 2015)	generative program	hierarchical parts	Bayesian prior	Bayesian-compression-like
LLM agent	Voyager (Wang+ 2023)	LLM / sim	LLM skills from feedback	none (ever-growing)	accumulation-without-pruning
LLM agent	DEPS (Wang+ 2023)	LLM / sim	LLM plan decomposition	none	accumulation
LLM agent	Reflexion (Shinn+ 2023)	LLM	stored verbal reflection	none (append)	accumulation-without-pruning
LLM agent	ExpeL (Zhao+ 2024)	LLM	extracted insights	weak heuristic	weak-keep

The pattern is not "everyone compresses." Compression (or a Bayesian prior that behaves like it) dominates grammar induction, program induction, and the recent HTN line; explicit utility appears in EBL; HRL is split, with most option-discovery using other objectives entirely; and the LLM/world-model agents mostly have no retention pressure. The claim is a disjunction: durable libraries need some retention pressure, and these are the recurring forms.

7 A Controlled Comparison: When Generalization Suffices, and When Specialization Pays

7.1 The synthetic comparison

The compression and utility branches are separate literatures. On a controlled domain they become directly comparable. Each candidate skill abstracts one task segment with two distinct properties: a frequency `f`, the fraction of tasks needing it, and a hardness `h`, the segment's blind-search cost (`B^h` model-rollouts if unabstracted). All segments share a description length, so an MDL retention rule's gain is proportional to `f` alone; compression is blind to hardness by construction, while a utility retention rule scores $f \cdot B^h$.

This is a mechanism demonstration, not an effect-size estimate: a space of any significant size needs both kinds of skill at once, the frequent general abstractions compression keeps and the rare specialists only a utility rule sees, so a criterion that prices one alone leaves the other half uncovered. Equal description lengths make MDL exactly frequency-ranking, which isolates the mechanism; in practice a skill's encoding length grows with its expansion, so real MDL is a noisy proxy for hardness rather than blind to it, and the correlation sweep below and Blocksworld restore the realistic case.

Bounded storage enters as a hard cap `K`. A bounded agent holds at most `K` skills, so a rule that keeps more is infeasible rather than merely costlier, though an admissible library still pays a per-step matching floor `B +`

|L|. Within the cap, MDL keeps the K most frequent skills, utility the K highest-$f \cdot B^h$. Expected held-out planning cost is exact, with no Monte-Carlo noise.

With B = 4, 30 candidate skills, and budget K = 10 (hard skills rare, ρ = −0.6):

retention rule	library size	held-out planning cost
no-library	0	10356
accumulate-all (unbounded ref)	30	265
frequency / MDL keep	10	8739
utility keep	10	831

No library is far worse, by orders of magnitude. Accumulate-all reaches 265 by keeping all thirty skills, but with |L| = 30 > K it breaks the storage bound; it is the unbounded-storage ceiling, a reference rather than a usable rule. Among rules that respect the cap, pruning alone does not fix the problem. MDL and a naive frequency cutoff barely improve on no library, spending the budget on frequent-but-easy skills and leaving the rare-hard segments uncovered. Only a utility rule recovers, keeping the abstractions that actually cut search, an order of magnitude below MDL and within roughly 3× of the unbounded ceiling at a third its storage. The retention criterion, not the act of keeping, carries the result.

7.2 The phase boundary

Whether that gap appears is conditional. Sweeping the correlation ρ between frequency and hardness against the budget K traces a phase boundary: utility's advantage runs up to roughly 28× where hard skills are rare and uncorrelated with frequency, and collapses toward parity as $\rho \to 1$, where the frequent skills are the hard ones and compression selects them anyway.

Bright = utility beats compression. The advantage fills the rare-and-uncorrelated regime and vanishes on the right, where frequency tracks hardness and compression picks the hard skills for free. Exact expected cost, averaged over 16 skill populations per cell.

So compression is a sound retention criterion exactly where statistical regularity tracks search value, and it fails precisely on the rare-but-critical abstraction it cannot see. Minton's utility problem and the MDL criterion are one picture: two prices on the same carrying cost, agreeing when frequency and difficulty align and diverging when they do not.

7.3 External validity

The same ablation in a standard planning domain confirms the boundary is real, not an artifact of the synthetic setup. In Blocksworld, with macro-operators mined from solved plans and planning cost measured as nodes expanded by greedy best-first search, the two retention rules nearly coincide and the gap widens over the tested sizes: utility beats MDL by 1.08× at five blocks, 1.18× at six, and 1.37× at seven, as deeper deadlocks make the rare search-saving macro matter more (mean expanded nodes with no library rise from 6.8 to 24.3 over the same range). Standard Blocksworld sits in the agreement corner, where frequency tracks search-value, and drifts toward divergence as the domain hardens. It is the kind of domain where a compression retention rule has served the symbolic lineages well. (Code, figure, and the Blocksworld harness: the `mine-then-keep` repository.)

A second confirmation comes from a deployed cognitive architecture, and it falls in the opposite corner. Derbinsky and Laird gave Soar's learned rules a forgetting mechanism, since shipped in the architecture as production apoptosis: a rule is excised when its base-level activation drops below threshold and it can be reconstructed by re-derivation (a reinforcement-tuned rule, whose learned value cannot be regenerated, is spared). Those are the two criteria exactly, activation the frequency (compression) side and reconstruction cost the miss-penalty (utility) side.

On Liar's Dice, where rare reinforcement-tuned rules carry most of the value, forgetting by activation alone collapses competence from 75% of games won to below 55%; adding the reconstruction-cost criterion restores it at a fraction of the memory. That is the divergence corner in a real architecture, the synthetic result reproduced where the stakes are higher: frequency-only retention evicts the specialists, and pricing their recomputation brings them back. Standard Blocksworld sits in the agreement corner and Liar's Dice in the divergence corner; together they bracket the phase boundary in deployed systems, not only the synthetic sweep.

8 Open Problems and Future Directions

The map makes three absences visible. The first is an empty column: every LLM and world-model skill-library agent in the taxonomy pairs the strongest proposal mechanism with no retention criterion at all. Voyager's library is ever-growing by design, Reflexion appends, and none weighs an abstraction's reuse value against its carrying cost. The agents with the richest world models are those with no mechanism for discarding what they learn, and a comprehensive survey of world models (ACM Computing Surveys, 2025) already names the abstract, high-level action layer as an open problem in its own right. That layer is a learned abstraction library: the field built the proposal half and left selection to accretion, with a recent wave of evolving skill graphs and self-improving libraries only beginning to treat management as a problem.

The field has already learned this lesson once. The utility problem Minton named in 1988, that a planner hoarding learned macros can spend more time matching them than they save, became a central reason macro-operator and explanation-based systems had to control what they kept. Soar's chunking could produce "expensive chunks" whose match cost degraded performance (Tambe, Newell, and Rosenbloom). And learned-method sets can still grow without obvious payoff: CURRICULAMA reports method count and planning time climbing in Blocks World, Logistics, and Rover. Each case is the same shape: unmanaged accumulation turning a learned library from an asset into a tax.

The result survives in the field's collective experience more than in any single paper's abstract, and accounts for much of why the older symbolic lineages developed a retention criterion at all: Soar itself, whose expensive chunks first showed the tax, later gained a forgetting mechanism that keeps the frequently used and the costly-to-reconstruct and discards the rest (Derbinsky and Laird). The skill-library agents have rebuilt the accumulation without that brake, and are positioned to relearn the result at scale.

The second absence is an untested assumption, and clearing it requires adjudicating two tempting readings. Compression rewards what recurs, but whether learned library items are reused at all is not automatic: a recent evaluation finds reported library-learning gains that trace to self-correction rather than reuse. Statistical regularity is a proxy for value, and the proxy can fail on its own terms. The second read is to take the bottleneck-option literature as already showing that rare skills carry outsized worth. It does not: diverse-density and betweenness subgoals are selected for sitting central or common on successful paths, so the work establishes leverage, not a rare-use, high-value separation. Only the line that prices planning time directly (Jinnai and colleagues) speaks to the frequency-value gap at all. So using bottleneck-option results as evidence for a rare-but-critical tail is a category slip; that research solves a different problem well.

The third absence is a methodological blind spot: the retention criterion is seldom ablated. The controlled comparison above is one of few, and it runs on a synthetic domain and Blocksworld, not a live agent.

These converge on one open problem, and it turns on a contingent fact: whether the value in an open-ended agent's skills concentrates in rare-but-critical abstractions, the heavy tail where compression and utility part. The structure is generic rather than exotic. A passport is used a few times a decade and is indispensable on each, and a rule that keeps what you use often deletes it first.

Value and frequency are independent axes, a point already settled for experience, where prioritized replay keeps the rare important transitions over the common ones, and settled earlier still in cognitive science: complementary learning systems theory (McClelland, McNaughton, and O'Reilly) holds that no single learner both extracts shared structure and retains sparse specifics, which is why the brain pairs a generalizing neocortex with a specific-storing hippocampus, the learning systems Kumaran, Hassabis, and McClelland argue an intelligent agent needs. The general library and the rare specialist are that pairing under a planning budget; planning utility is the retention criterion that catches the specialist, the objective Jinnai and colleagues optimize directly.

Whether a real agent's library inherits this shape is unmeasured, so we leave it open. But the analogy fixes the prior: a planner whose hardest sub-problems are rare bottlenecks, the one deadlock-breaking maneuver, the unlock that opens a region, learns its highest-value skills exactly where compression is blind. If so, the agents accumulating libraries today are keeping the wrong half.

Three steps would close it: extend the controlled comparison to a more combinatorial domain such as Logistics, and to larger Blocksworld, to trace the full agreement-to-divergence drift; measure the value-versus-frequency distribution of a real skill-library agent's abstractions directly; and ablate retention criteria in a live world-model agent over a long horizon. Our own grid-puzzle agent, a learned simulator with a decomposition library grown by proposal-then-compression, is the intended vehicle for the last.

9 Conclusion

Two communities that rarely cite each other have been computing nearly the same library: generalization and specialization, under the names compression and utility, coincide wherever reuse tracks difficulty, the regime standard Blocksworld occupies, where either rule keeps the same library and each field succeeds with its own. They part only on the rare-but-critical abstraction, the corner we conjecture the newest agents occupy and the one their accreting libraries cannot keep. This map is one lens among others, and a survey's framing can ossify a field as easily as clarify it; the lens earns its place only if it makes the retention criterion visible as a choice rather than an accident, and names the single measurement that would settle which choice long-horizon agents need.

10 References

- Anderson, Schooler. Reflections of the Environment in Memory. Psychological Science 2(6), 1991.
- Bacon, Harb, Precup. The Option-Critic Architecture. AAAI, 2017.
- Belady. A Study of Replacement Algorithms for a Virtual-Storage Computer. IBM Systems Journal 5(2), 1966.
- Berlot-Attwell, Rudzicz, Si. Library Learning Doesn't: The Curious Case of the Single-Use 'Library'. 2024.
- Bowers, Olausson, Wong, Grand, Tenenbaum, Ellis, Solar-Lezama. Top-Down Synthesis for Library Learning (Stitch). POPL, 2023.
- Cao, Irani. Cost-Aware WWW Proxy Caching Algorithms (GreedyDual-Size). USITS, 1997.
- Castel, Benjamin, Craik, Watkins. The Effects of Aging on Selectivity and Control in Short-Term Recall. Memory & Cognition 30(7), 2002.
- Cherkasova. Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy (GDSF). HP Labs TR, 1998.
- DeJong, Mooney. Explanation-Based Learning: An Alternative View. Machine Learning 1(2), 1986.
- Derbinsky, Laird. Effective and Efficient Forgetting of Learned Knowledge in Soar's Working and Procedural Memories. Cognitive Systems Research 24, 2013.
- Ellis, Wong, Nye, Sablé-Meyer, Morales, Hewitt, Cary, Solar-Lezama, Tenenbaum. DreamCoder: Growing Generalizable, Interpretable Knowledge with Wake-Sleep Bayesian Program Learning. 2021.
- Eysenbach, Gupta, Ibarz, Levine. Diversity Is All You Need: Learning Skills without a Reward Function (DIAYN). ICLR, 2019.
- Fikes, Hart, Nilsson. Learning and Executing Generalized Robot Plans (STRIPS MACROPS). Artificial Intelligence 3, 1972.
- Hérail, Bit-Monnot. Learning Operational Models from Demonstrations: Parameterization and Model Quality Evaluation. ICAPS HPlan Workshop, 2022.
- Hérail, Bit-Monnot. Leveraging Demonstrations for Learning the Structure and Parameters of Hierarchical Task Networks. FLAIRS, 2023.
- Hérail. Learning Hierarchical Models from Demonstrations for Deliberate Planning and Acting. PhD thesis, Université de Toulouse, 2024.
- Hogg, Muñoz-Avila, Kuter. HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. AAAI, 2008.
- Ilghami, Nau, Muñoz-Avila, Aha. CaMeL: Learning Method Preconditions for HTN Planning. AIPS, 2002.
- Jiang, Liu, Eysenbach, Kolter, Finn. Learning Options via Compression (LOVE). NeurIPS, 2022.
- Jinnai, Abel, Hershkowitz, Littman, Konidaris. Finding Options that Minimize Planning Time. ICML, 2019.
- Konidaris, Barto. Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining. NeurIPS, 2009.
- Kumaran, Hassabis, McClelland. What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated. Trends in Cognitive Sciences 20(7), 2016.
- Laird, Rosenbloom, Newell. Chunking in Soar: The Anatomy of a General Learning Mechanism. Machine Learning 1, 1986.
- Lake, Salakhutdinov, Tenenbaum. Human-Level Concept Learning through Probabilistic Program Induction (BPL). Science 350, 2015.
- Lam, Mörchen, Fradkin, Calders. Mining Compressing Sequential Patterns (GoKrimp). Statistical Analysis and Data Mining 7(1), 2014.

- Larsson, Moffat. Off-Line Dictionary-Based Compression (RePair). Proceedings of the IEEE 88(11), 2000.
- Li, Nau, Roberts, Fine-Morris. Automatically Learning HTN Methods from Landmarks (CURRICULAMA). 2024.
- Machado, Bellemare, Bowling. A Laplacian Framework for Option Discovery in Reinforcement Learning (eigenoptions). ICML, 2017.
- McClelland, McNaughton, O'Reilly. Why There Are Complementary Learning Systems in the Hippocampus and Neocortex. Psychological Review 102(3), 1995.
- McGovern, Barto. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. ICML, 2001.
- Minton. Quantitative Results Concerning the Utility of Explanation-Based Learning (PRODIGY). AAAI, 1988.
- Nejati, Langley, Könik. Learning Hierarchical Task Networks by Observation. ICML, 2006.
- Nevill-Manning, Witten. Identifying Hierarchical Structure in Sequences: A Linear-Time Algorithm (SEQUITUR). JAIR 7, 1997.
- Pickett, Barto. PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning. ICML, 2002.
- Schaul, Quan, Antonoglou, Silver. Prioritized Experience Replay. ICLR, 2016.
- Shinn, Cassano, Berman, Gopinath, Narasimhan, Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. NeurIPS, 2023.
- Şimşek, Barto. Skill Characterization Based on Betweenness. NeurIPS, 2008.
- Solan, Horn, Ruppin, Edelman. Unsupervised Learning of Natural Languages (ADIOS). PNAS 102(33), 2005.
- Sutton, Precup, Singh. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning (options). Artificial Intelligence 112, 1999.
- Tambe, Newell, Rosenbloom. The Problem of Expensive Chunks and Its Solution by Restricting Expressiveness. Machine Learning 5, 1990.
- Understanding World or Predicting Future? A Comprehensive Survey of World Models. ACM Computing Surveys, 2025.
- Wang, Xie, Jiang, Mandlekar, Xiao, Zhu, Fan, Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models. 2023.
- Wang, Cai, Liu, Ma, Liang. Describe, Explain, Plan and Select (DEPS). NeurIPS, 2023.
- SkillGraph: Skill-Augmented Reinforcement Learning for Agents via Evolving Skill Graphs. 2026.
- Zhao, Huang, Xu, Lin, Liu, Huang. ExpeL: LLM Agents Are Experiential Learners. AAAI, 2024.