CARL: Chat Assisted Reinforcement Learning

Sanjiv Das, Harshad Khadilkar, Sukrit Mittal, Daniel Ostrov, Jacob Sisk

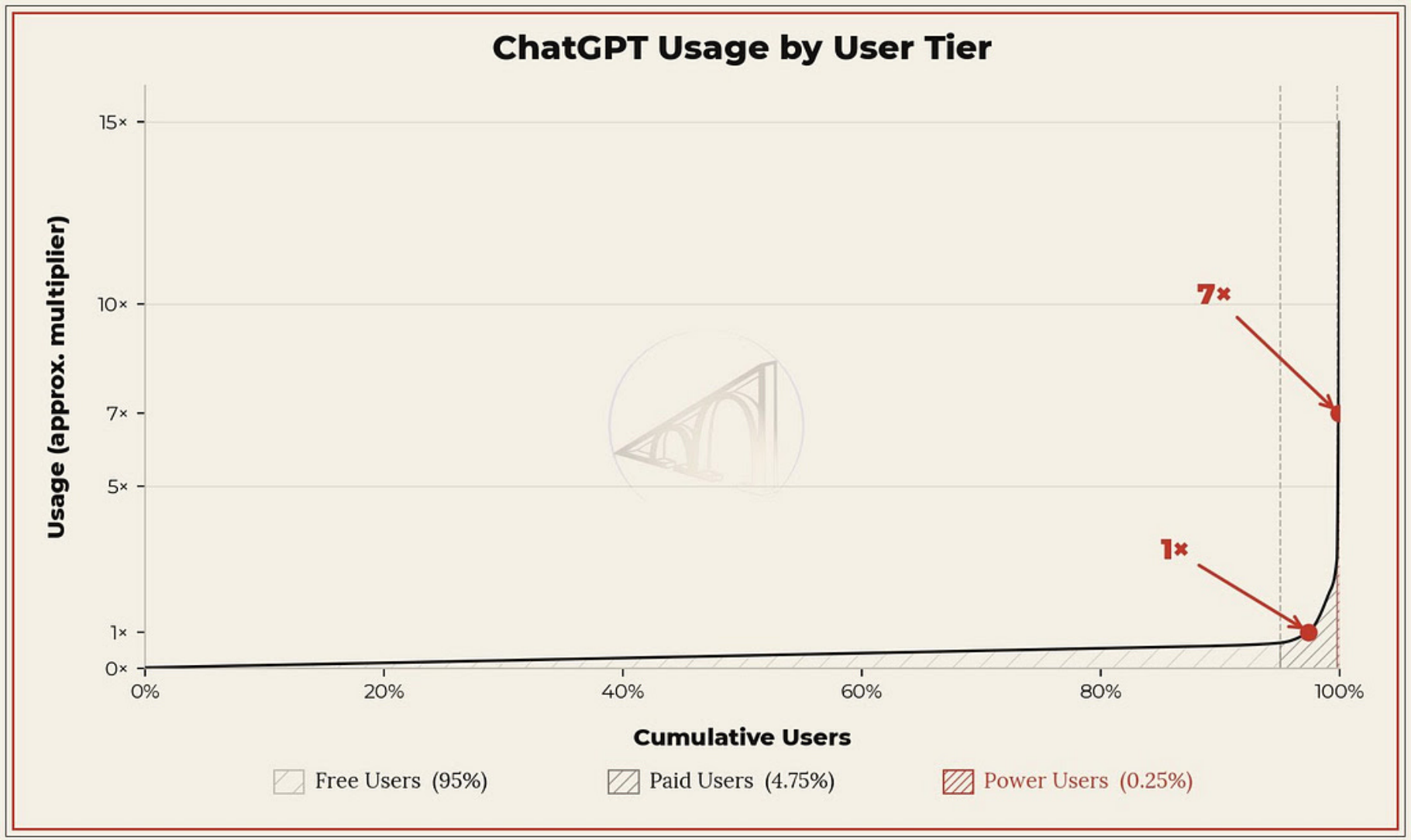
Santa Clara University & Franklin Templeton

@JOIM, Singapore, November 2026

Outline

1. Reinforcement learning for wealth management problems and far more
2. Using agents to democratize RL (Reinforcement Learning) solutions for wealth management
3. The story: A traversal from Reinforcement learning (RL) $\rightarrow$ MetaRL $\rightarrow$ CARL
4. Insight: General agent harnesses cannot do it all – we do need special harnesses with built-in domain expertise.

Capability Overhang



Capability Overhang

1. Most people use Chat, etc., as better search. They're missing 99% of the value.
2. The AI companies call this a capability overhang. The models have been able to do far more than anyone uses them for, but nobody noticed because the tooling hadn't caught up. But now we are democratizing Agentic AI.
   https://cdn.openai.com/pdf/openai-ending-the-capability-overhang.pdf
3. The number of tokens used by agents has now surpassed the number used by humans.

Components of Agentic AI Infrastructure

- **Models** are the Core "Brains": These underlying systems, such as Claude-Opus-4.8, Gemini 3 Pro, or GPT-5.5, determine the platform's actual intelligence and reasoning capabilities. They are the engine behind the scenes that dictates how effectively the AI can write, code, or analyze complex data sets.
- **Apps** are the User Interface: Apps are the specific products and websites, like chatgpt.com or Claude Cowork, that allow you to interact with the underlying models. They act as the bridge between the user and the AI, providing a functional space to perform tasks on a phone, desktop, or browser.
- **Harnesses** Enable Action: A harness enables an AI model to execute multi-step workflows in a aelf-managed loop. While apps often include a built-in harness, standalone versions like OpenClaw allow models to take direct action on a computer to complete complex projects from scratch.
- **Agent**: A model with a harness and tools, skills, etc.

What has changed?

1. Models escaped the chat box. The biggest rate limiter to AI value hasn't been intelligence — it's been the interface. That's over. Claude is now working directly in Excel, Notion, Linear, Slack — not as a sidebar, but as the analyst.
2. The models keep getting better — and are improving themselves. – OpenAI used early Codex to debug its own training runs. The model helped build the model. Better models → better tools → better models. The flywheel is rapid (RSI).
3. Task length has exploded. Research from METR shows autonomous task horizons doubling every four months. At 30 minutes: code snippets. At 5 hours: module refactoring. At multi-day: full codebase audits. Each doubling unlocks exponentially more of the total automation pie.
4. Agents work in teams. Cursor built an agent harness that orchestrates thousands of agents simultaneously. Their FastRender experiment had 30,000 commits, 2,000 concurrent agents, and a million+ lines of Rust. With a planner/worker architecture with specialized roles.

Two Very Different RL Optimization Examples

Goals-Based Wealth Management (GBWM)

Investors have multiple goals they want to attain like...

- Purchasing a Car
- Going on a Trip
- Having enough money for Retirement
- Giving money to Charity
- Etc.

Each goal has three attributes:

1. The time at which the goal is available. (E.g., the investor wants a car five years from now.)
2. The cost of attaining the goal. (E.g., the car will cost $40,000.)
3. The importance of the goal to the investor (measured by the utility of the goal to the investor).

Optimizing GBWM Solutions

Optimization Objective

Investors want to optimize the number of their attained goals, weighted by their importance. That is, we look to maximize the expected value of the sum of the attained goals' utilities.

Actions to Optimize

To do this, we must determine the investor's optimal dynamic strategy, meaning...

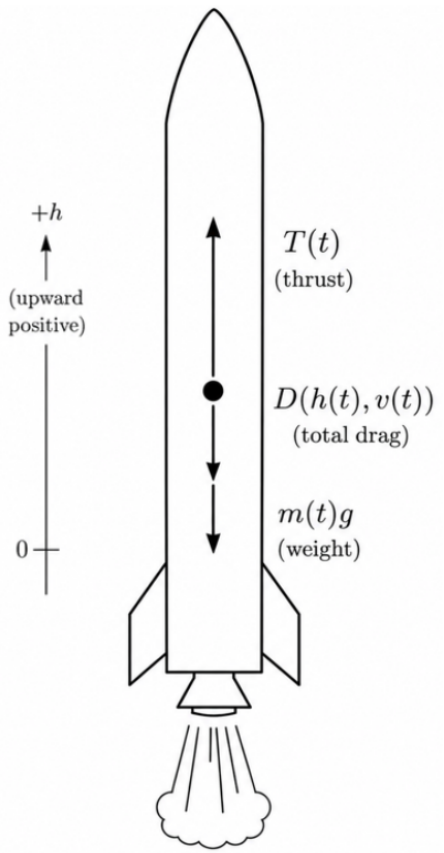
1. determining the optimal investment portfolio to be in now, knowing it can change each year in the future.
2. determining the optimal goal-taking choice for a currently available goal, meaning whether it is best to attain the goal (and its utility) or forgo the goal (to save money, increasing the probabilities for attaining later goals).

Top 10 RL use cases in finance

1. Optimal Trade Execution (Order Slicing)
2. Dynamic Portfolio Rebalancing
3. Market Making and Inventory Management
4. Continuous-Time Options Hedging
5. Algorithmic High-Frequency Trading (HFT)
6. Dynamic Risk Management & Stress Testing
7. Automated Credit Scoring & Loan Pricing
8. Crypto Yield Farming & DeFi Liquidity Routing
9. Robo-Advising & Goal-Based Wealth Planning
10. Fraud Detection & Anti-Money Laundering (AML)

...and finance is just one field that RL can be applied to.
Our next example is from the field of engineering...

Goddard Rocket Problem



Rocket fuel is used to provide thrust, $T(t)$, to the rocket, which must overcome the rocket's weight and the air drag on the rocket.

Optimization Objective

Maximize the highest altitude, h, attained by the rocket.

Action to Optimize

To do this, we must determine the optimal thrust $T(t)$ to apply over time, given there is a limit, $T_{\max}$, on the thrust and a finite amount of fuel.

Goddard Rocket Problem Solution

The rocket hull can only withstand so much pressure, so there is a dynamic pressure constraint: $\frac{1}{2}\rho(h(t))|v(t)|^2 \leq Q_{max}$.

If there is no constraint ($Q_{max} = \infty$)...

The three phase **optimal** solution is **known**:

1. First phase: start with full thrust $T(t) = T_{\max}$.
2. Second phase: immediately lower thrust and then follow a known thrust function over time until fuel runs out.
3. Third phase: coast until the rocket hits its highest altitude.

If there is a constraint (Q_{max} is finite)...

The optimal solution is **not known**.
RL is an excellent technique for trying to determine the optimal solution!

Primer: Reinforcement Learning

Basis: Why RL?

- Optimal solution is:
  1. Not possible to define in algebraic terms
  2. Intractable to compute
  3. Too compute-intensive to solve for
- In the case of supervised learning (e.g. regression), we:
  1. Provide examples: situations x vs. solutions y
  2. Fit a function approximator, e.g. $y = mx + c$
  3. Guess the solutions for unseen situations
- When examples of 'good' solutions cannot be provided (e.g. you can *evaluate* a solution but you don't know how to improve it), we use RL

Primer: Reinforcement Learning

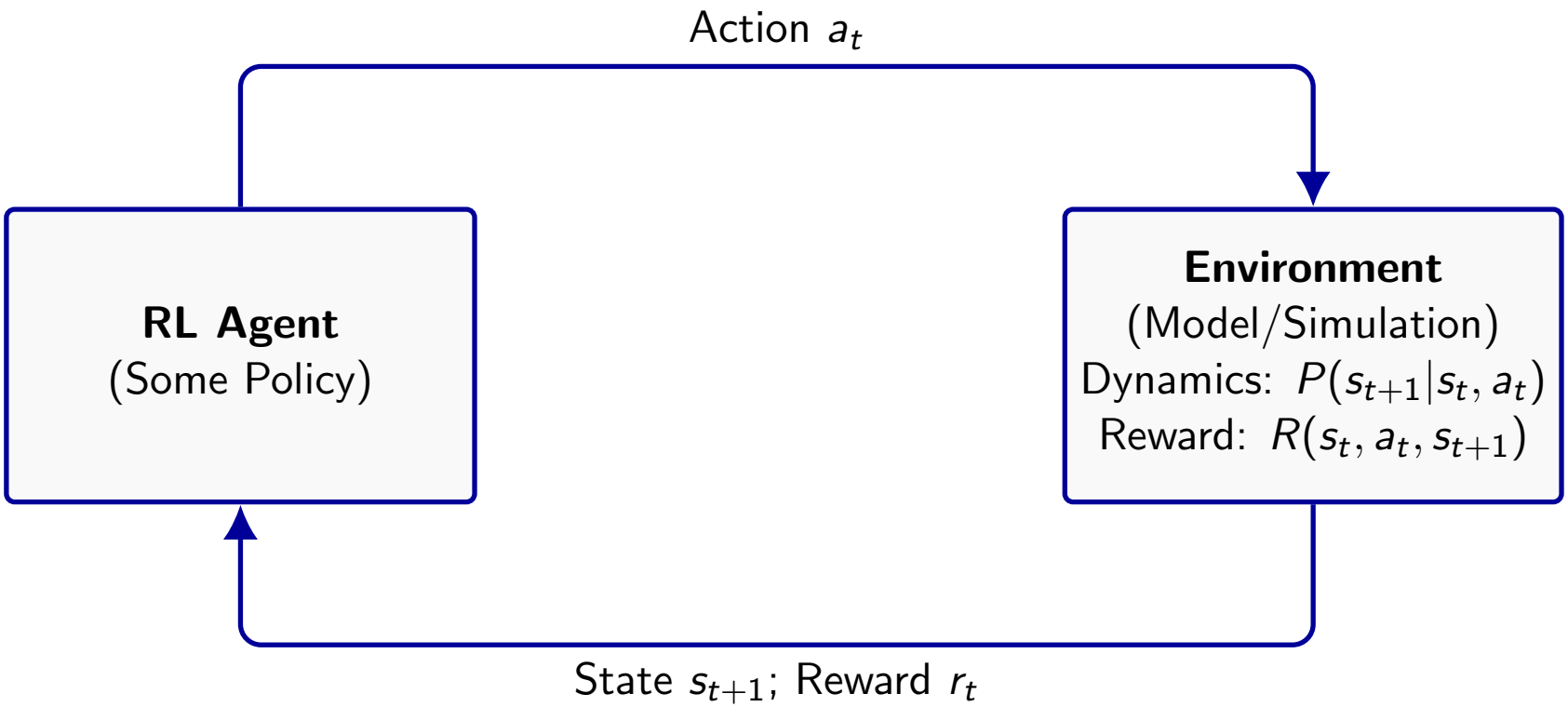


Figure: The agent-environment interaction loop in Reinforcement Learning.

The policy $\pi : s_t \rightarrow a_t$ is implemented by neural network(s)

Solving GBWM with RL (arxiv 2605.02300)

What

- Goal-based wealth management with multiple goals (defined by cost and utility)
- Pre-trained Meta-model (MetaRL) model computes portfolio and goal-taking decisions purely in inference mode (*zero-shot*), without requiring online training

Why

- Inference is fast: 10 ms per decision on average
- Approach is scalable to large state spaces
- It is economical to train, in a few hours
- Pure inference accuracy is $\approx 97.8\%$ on average, and could potentially be further improved with fast online fine-tuning

Hierarchical Actor-Critic Architecture

GoalAgent and PortfolioAgent are independent but sequential entities, where the output of the first influences the environment or state for the second.

Category	**Actor Network**	**Critic Network**
Input Dimension	26 ($5 + 3K$, $K = 7$)	26 ($5 + 3K$, $K = 7$)
Hidden Layers (Nodes)	256, 64, 16	64, 16
Activation (Intermediate)	`tanh`	`tanh`
Activation (Output)	`sigmoid`	`linear`
Output Type	Scalar (a_g or a_p)	Scalar (Value Function)
Hyperparameter	**Value**	
Learning Rate	10^{-4}	
PPO Clip Parameter (ϵ)	0.2	
Discount Rate (γ)	1.0	
Training Epochs	1000	
Episodes per Epoch	500	
Total Training Time	$\approx$ 4 hours	

Sample training result

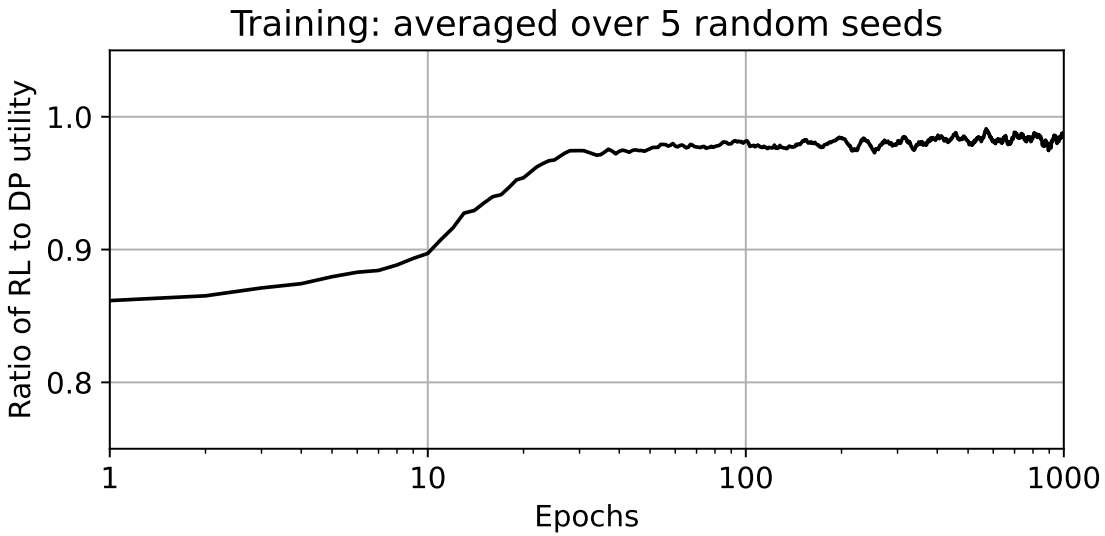


Figure: Performance during training, 1000 randomly generated problems averaged over 5 random seeds, with horizons up to 50 years. Training for each of the 1000 problems is done over 500 epochs. Shaded region indicates standard deviation.

The performance, denoted by the ratio of RL to DP utility, is fairly stable after 100 epochs

Case Studies: Samples

1. Single-goal: Planning to purchase a house in 10 years.
2. Multi-goal: Planning to purchase a car in 3 years, Europe vacation in 5 years, and house in 15 years.
3. Retirement-goal: Planning the annual income for the retirement tenure, till life expectancy.

- Testing horizons go up to 100 years to evaluate out of distribution capability.
- Number of goals varies from 1 to 60 with a mean of 38.

Table 9: Description of test cases with infusions. The base amount of infusion for odd-numbered test cases is $\tilde{I} = \frac{W(0)}{10(T-1)}$, rounded up to the nearest integer.

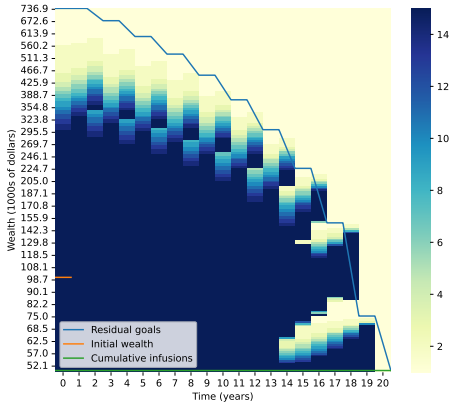
Case	T	$W(0)$	Non-zero goals $[t, C(t), U(t)]$	Non-zero infusions $[t, I(t)]$
34	10	100	$[10, 150, 1]$	$[1, 10]$
35	10	100	$[10, 200, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
36	10	100	$[10, 400, 1]$	$[1, 10]$
37	40	100	$[40, 600, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
38	40	100	$[40, 1200, 1]$	$[6, 12]$
39	40	100	$[40, 2400, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
40	100	100	$[100, 50000, 1]$	$[21, 19]$
41	100	100	$[100, 500000, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
42	100	100	$[100, 2000000, 1]$	$[27, 22]$
43	3	100	$[2, 75, 0.9], [3, 75, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
44	20	100	$[10, 200, 1.3], [20, 500, 1]$	$[6, 12]$
45	20	100	$[15, 200, 1.3], [20, 300, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
46	35	100	$[5, 50, 1], [25, 500, .5], [35, 1000, 1]$	$[13, 15]$
47	60	100	$[15, 300, .7], [30, 6000, 1.2],$ $[45, 5000, .2], [60, 20000, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
48	25	100	$[3, 30, .2], [5, 70, .3], [8, 70, .3],$ $[25, 1000, 1]$	$[11, 14]$
49	40	100	$[10, 150, 1.5], [30, 400, 1],$ $[35, 500, 1], [40, 600, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
50	20	100	$[t, 15, 1]$ for $t \in \{2, 4, 6, \ldots, 20\}$	$[10, 13]$
51	20	100	$[t, 25, 1]$ for $t \in \{2, 4, 6, \ldots, 20\}$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
52	20	100	$[t, 50, 1]$ for $t \in \{2, 4, 6, \ldots, 20\}$	$[11, 14]$
53	20	100	$[t, 75, 1]$ for $t \in \{2, 4, 6, \ldots, 20\}$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
54	60	100	$[t, 20, 1]$ for $t \in \{1, 2, 3, \ldots, 60\}$	$[38, 31]$
55	60	100	$[t, t, 1]$ for $t \in \{1, 2, 3, \ldots, 60\}$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
56	60	100	$[t, t, 100 + t]$ for $t \in \{1, 2, 3, \ldots, 60\}$	$[41, 34]$
57	60	100	$[t, t, 100 - t]$ for $t \in \{1, 2, 3, \ldots, 60\}$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
58	60	100	$[t, 60 - \frac{t}{2}, 1]$ for $t \in \{1, 2, 3, \ldots, 60\}$	$[45, 38]$
59	60	100	$[t, 60 - \frac{t}{2}, 100 + t]$ for $t \in \{1, 2, \ldots, 60\}$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
60	60	100	$[t, 60 - \frac{t}{2}, 100 - t]$ for $t \in \{1, 2, \ldots, 60\}$	$[49, 43]$
61	30	100	$[3, 35, 1.5], [6, 35, 1.3], [9, 5, 0.4],$ $[12, 50, 1], [15, 15, 0.7], [18, 5, 0.3],$ $[21, 45, 0.6], [24, 120, 0.9], [27, 170, 1.1],$ $[30, 160, 1]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
62	16	12	$[16, 34.25, 26]$	$[14, 3]$
63	16	21.63	$[8, 18.50, 18], [16, 34.25, 26]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
64	16	38.99	$[4, 13.60, 14], [8, 18.50, 18],$ $[12, 25.18, 22], [16, 34.25, 26]$	$[15, 6]$
65	16	70.27	$[2, 11, 66, 12], [4, 13.60, 14],$ $[6, 15.87, 16], [8, 18.50, 18],$ $[10, 21.59, 20], [12, 25.18, 22],$ $[14, 29.37, 24], [16, 34.25, 26]$	$[t, 1.03^t \tilde{I}]$ for $t \in \{1, .., T-1\}$
66	16	126.67	$[1, 10.8, 11], [2, 11, 66, 12],$ $[3, 12.60, 13], [4, 13.60, 14],$ $[5, 14.69, 15], [6, 15.87, 16],$ $[7, 17.14, 17], [8, 18.50, 18],$ $[9, 19.99, 19], [10, 21.59, 20],$ $[11, 23, 32, 21], [12, 25.18, 22],$ $[13, 27.20, 23], [14, 29.37, 24],$ $[15, 31.72, 25], [16, 34.25, 26]$	$[16, 21]$

Case 20: DP/RL vs. MetaRL

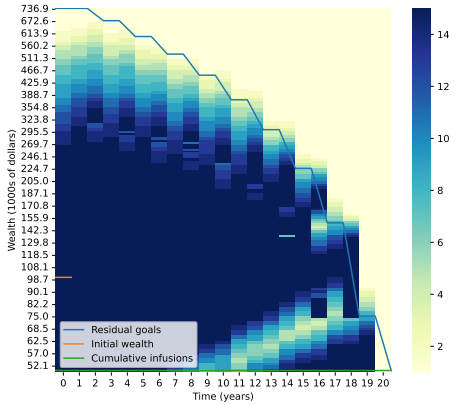
Scenario details:

1. $T = 20$ years
2. $W_0 = \$100$
3. Goals:
   1. $t \in \{2, 4, 6, \ldots\}$
   2. $C(t) = \$75$
   3. $U(t) = 1$

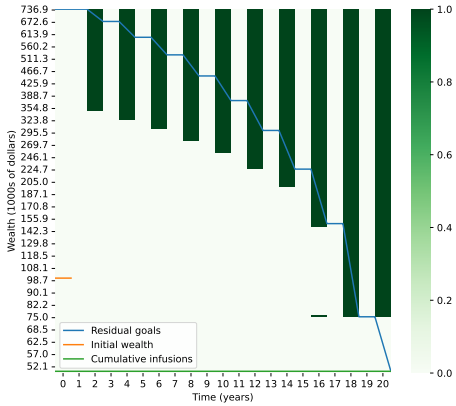
DP/RL: Portfolio policy



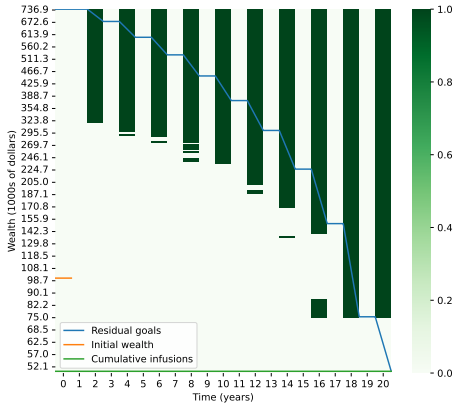
MetaRL: Portfolio policy



DP/RL: Goal-taking policy



MetaRL: Goal-taking policy

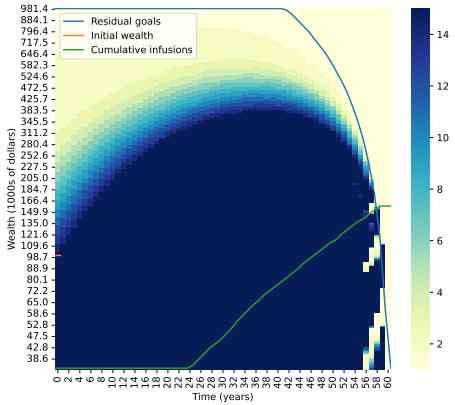


Case 57: DP/RL vs. MetaRL

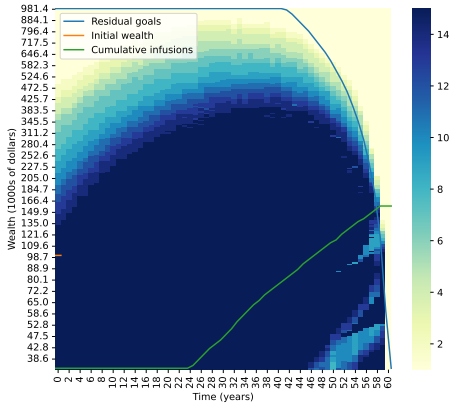
Scenario details:

1. $T = 20$ years
2. $W_0 = \$100$
3. Goals:
   1. $t \in \{1, 2, 3, \ldots\}$
   2. $C(t) = \$t$
   3. $U(t) = 100 - t$
   4. total infusions of \$157 spread uniformly across the tenure.

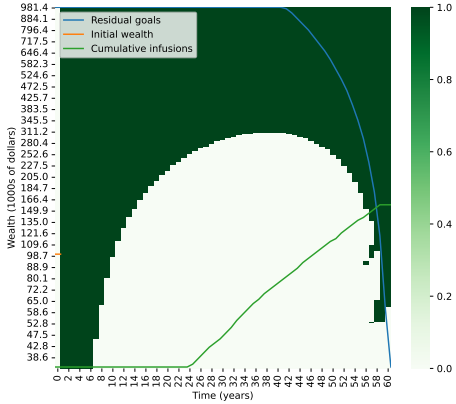
DP/RL: Portfolio policy



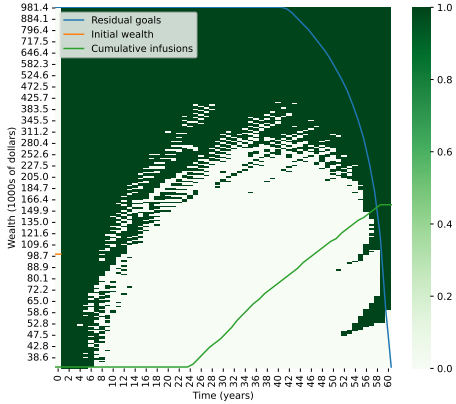
MetaRL: Portfolio policy



DP/RL: Goal-taking policy



MetaRL: Goal-taking policy



Results Summary

Time (in ms)	Mean	Std. dev.	Min.	Median	Max.
MetaRL inference	20.94	1.34	16.5	21.1	23.9
DP solution	2198	2255	83	1248	8012
Speed-up	$\approx$ 105x		$\approx$ 5x	$\approx$ 59x	$\approx$ 335x

Observations

1. In finding the optimal actions at current time, RL may provide upto 335x speed-up compared to RL/DP, depending on the scenario.
2. This is a significant saving in time as well as computational resource, given tens of thousands of investors could be served.

But we did this old-school

- Hand-built simulation
- Hand-written PPO optimizer for the RL agents
- $\approx$ 6 months of development time

Can an AI agent construct a full reinforcement learning (RL) system from an ambiguous business objective?

Can an AI agent construct a full reinforcement learning (RL) system from an ambiguous business objective?

No.

Can an AI agent, together with a human user, construct a full reinforcement learning (RL) system from an ambiguous business objective?

Can an AI agent, together with a human user, construct a full reinforcement learning (RL) system from an ambiguous business objective?

YES!

Three Pillars of Applied RL

- **Domain Expertise**: Understanding the specific problem domain and business context.
- **Problem Definition, MDP Design**: Structuring the core Markov Decision Process (MDP) by explicitly defining states, actions, rewards, and dynamics.
- **System Implementation**: Executing a runnable code repository to handle reward shaping, hyperparameter tuning, environment validation, model training, and final policy evaluation.

Chat Assisted Reinforcement Learning (CARL)

Paper: arxiv2610.31014

The Core Problem: The **Specification Gap**
CARL bridges the specification gap in complex applied RL.

- Implementation Stage: Modern agentic systems excel at coding and hyperparameter tuning. However, autonomous code generation addresses only the execution stage.
- Specification Stage: Failures often originate earlier, when human intent has not yet been translated into a precise problem definition or Markov Decision Process (MDP).

Design Principle: A correctly specified problem with a suboptimal algorithm can be improved incrementally; a correct implementation of the wrong MDP optimizes the wrong problem.

The RL Specification Gap & LLMs

- **The Root Failure:** Applied RL projects primarily fail due to underspecified problem definitions, not algorithmic shortcomings (Hamadanian, 2022).
- **The Tool Gap:** Implementation suites (e.g., `Stable-Baselines3`, `FinRL`) inherently assume formal MDP/reward structures are already complete (Schulman et al., 2017; Liang et al., 2021; Liu et al., 2022).
- **LLM Limitations:** Standard assistants (e.g., `ChatGPT`) lack structured step-by-step design logic (Bai et al., 2024), leading to execution anomalies like *reward hacking* and *state aliasing*.

Logical Design of CARL

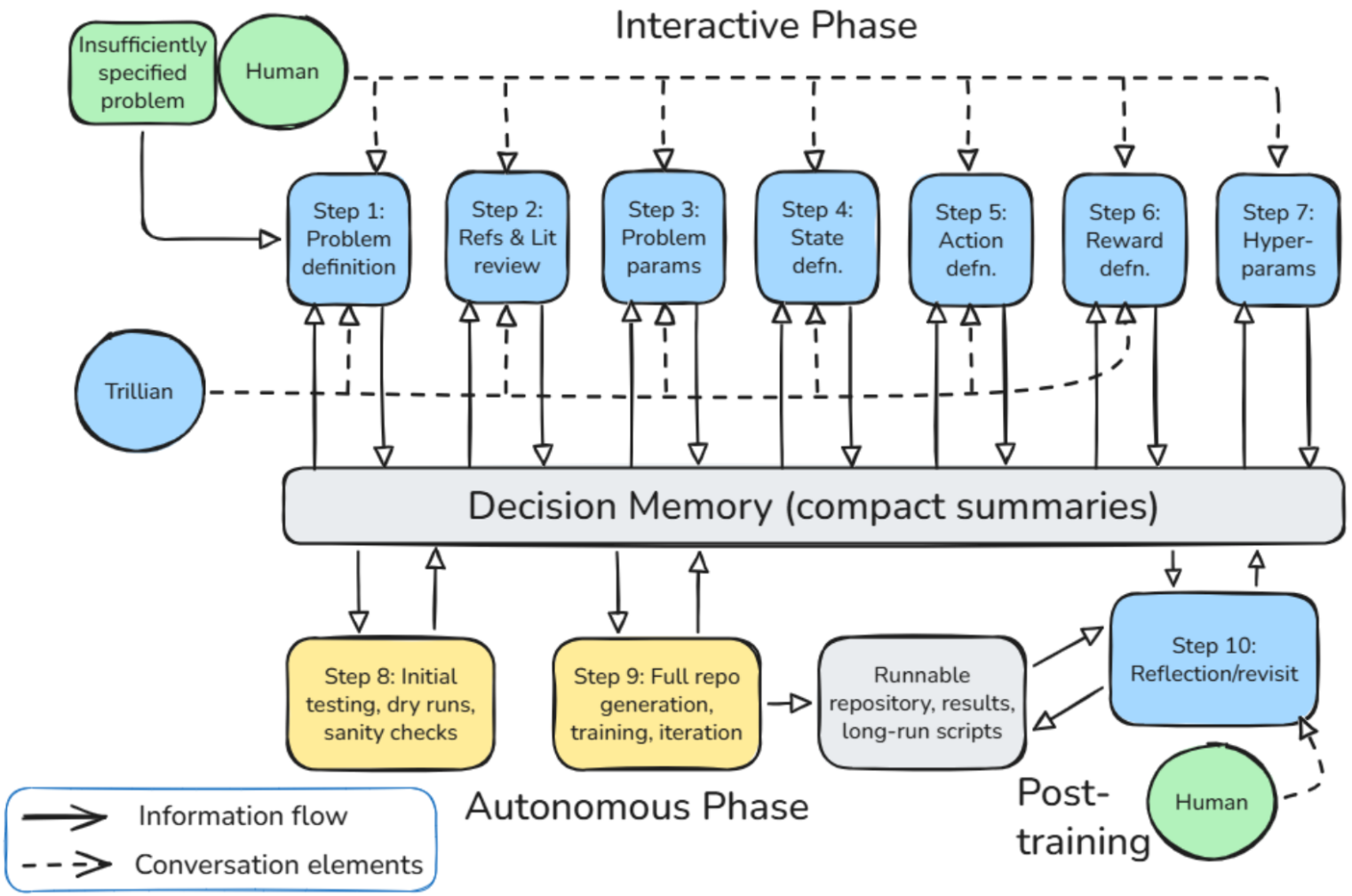


Figure 1 **Logical design of CARL, showing the interactive specification phase, autonomous implementation phase, Trillian reply assistance, and post-training revisit loop.**

Ten Step Workflow

Step	Skill	Objective	Approx. tokens
1	Problem Definition	Clarify domain objective, dynamics, constraints	900
2	References	Gather domain knowledge, data sources, baselines	350
2a	Online Search (**Optional**)	Search for related literature to assist formulation	600
3	Parameter Exploration	Define exogenous variables and randomization ranges	700
4	Information Proposal	Specify state variables and normalization	1,150
5	Action Definition	Specify action space and algorithm family	1,250
6	Reward Definition	Specify objective, reward, and shaping logic	1,250
7	ML/RL Parameters	Confirm learning settings and training budget	2,300
8	Initial Testing	Validate generated artifacts against the specification	2,200
9	Launch Training	Train, evaluate, and summarize the learned policy	4,450
10	Revisit (**Optional**)	Analyze results, edit MDP or hyperparameters, retrain	5,500

Table 1 CARL logical steps. Steps 1–7 are interactive specification steps; Steps 8 and 9 are autonomous implementation steps governed by the accumulated specification. Step 10 can be launched by the user if they are dissatisfied with the results from previous runs and wish to run another iteration. (New iterations can be done an arbitrary number of times.) CARL itself recommends whether a revisit and new iteration should occur at the end of Step 9. Token counts are approximate full-skill-file prompt sizes, rounded from a character-based estimate.

Problem Definition

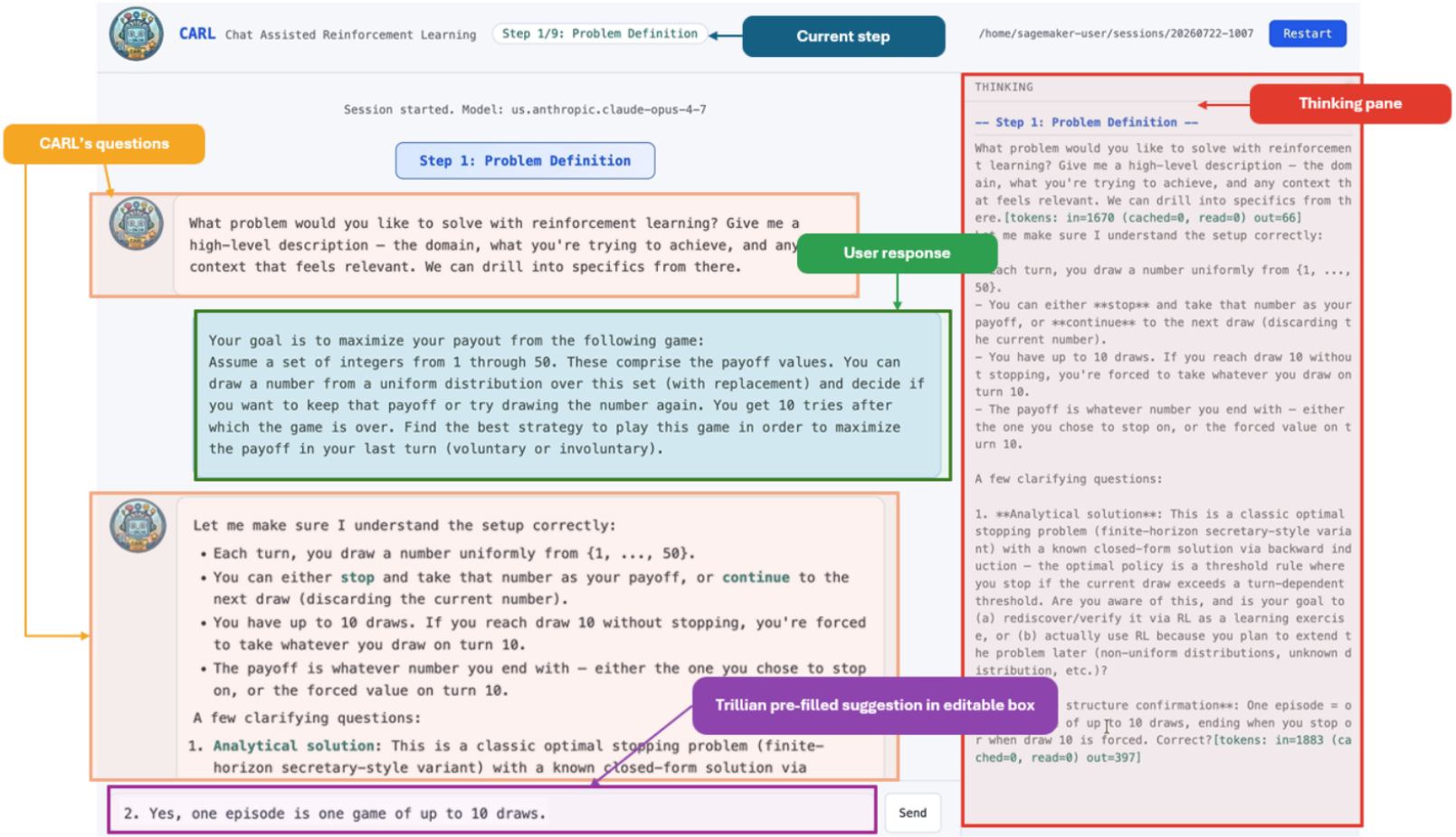


Figure 2 **[Time stamp 00:15] CARL web interface captured during Step 1 for our sample problem. CARL asks some clarifying questions to ensure that the rules of the game are clearly understood (better than the 2-3 sentence description initially provided). Trillian suggests answers that the user can choose to edit.**

Step 6 (Reward Definition) Skill

Condensed excerpt from the Step 6 skill, with a local objective, design hints, and checklist items.

Skill element	**Excerpt from Step 6: Reward Definition**
Description	Design the reward with normalization and domain randomization; ask for hints or nudges for early training.
Design hints	Keep the reward a single readable expression; drop fixed constants; expose reward components in `info`; document discounting.
Reward checks	Complete reward-configuration and credit-coverage tables; ensure terminal/failure terms reach responsible agents; label unmatched oracle formulas as heuristic.
Randomization checks	Randomize policy-relevant parameters at reset; include randomized parameters in the observation; disable randomization during evaluation.

Step 7

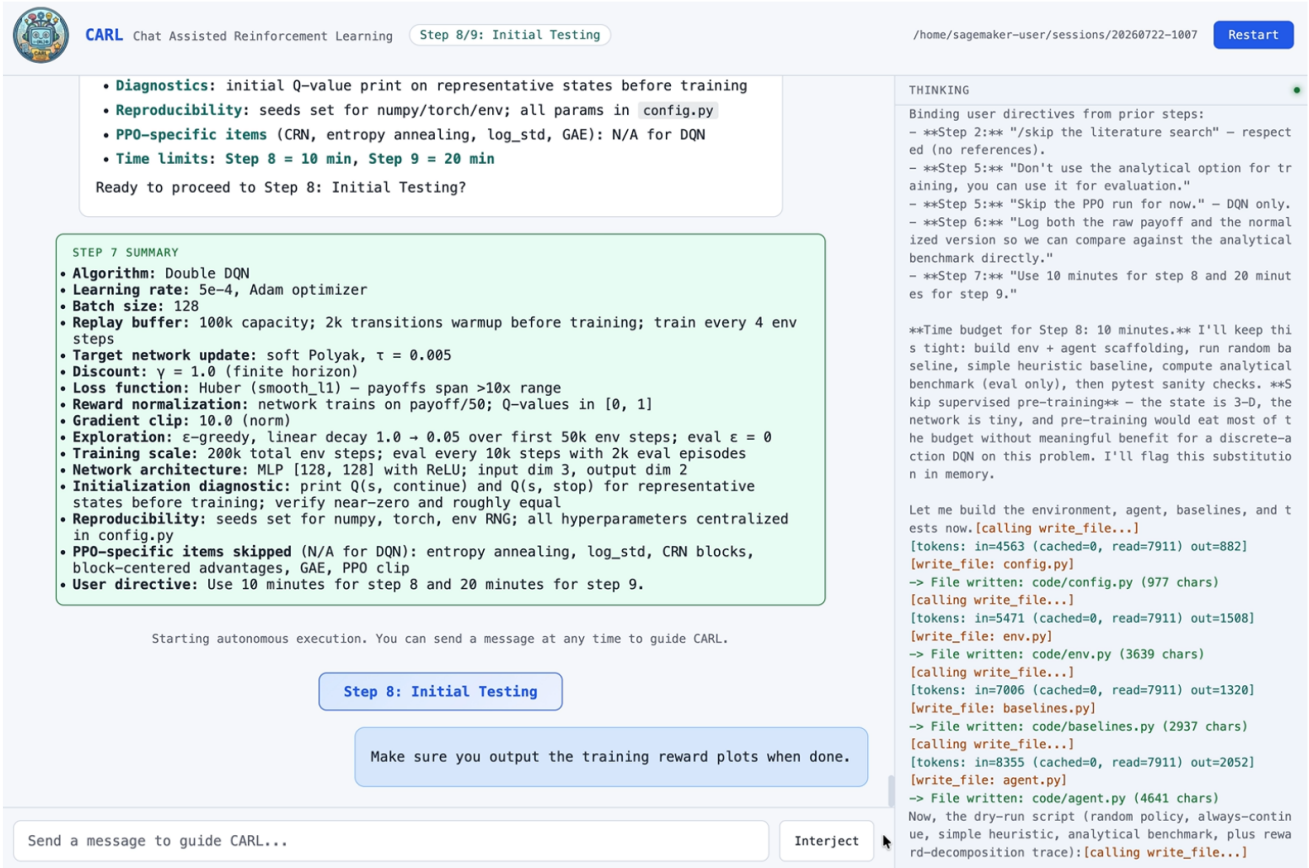


Figure 3 **Sample decision summary (in green box) which is added to `memory.md`. This is the summary after Step 7 for the example problem from Subsection 3.2 (time stamp 14:10). To improve context management, only the set of compact summaries from previous steps are fed to future steps. An interjection from the user is also visible at the bottom, with an instruction on the form of reporting at the end. In autonomous mode, such instructions are used for steering CARL mid-flow.**

Demo: https://youtu.be/Qx673uczp4c

Auxilliary Features

- **Brainstorming mode**: Activates via `/suggest` in Step 1 to search literature and synthesize candidate formulations for users with only broad domain interests. The chosen formulation becomes a reference document routed into the standard workflow.
- **Step 10 revisit mode**: Reopens completed sessions for post-training diagnosis by reviewing memory, logs, plots, and code. Proposed changes are discussed interactively and prepared in a fresh workspace rather than altering the completed run.
- **Autonomous-step interjections**: Allows users to inject guidance during the autonomous generation, testing, or training phases. Messages are added to the stream to redirect implementation without breaking step boundaries or restarting the workflow.
- **LaTeX report output**: Compiles a paper-style document from accumulated design memory, execution artifacts, logs, plots, and literature once a session completes. Serves as official workflow documentation rather than an active part of the MDP design.

CARL: Six Core Design Principles (Section 4.1)

1. **Separate Specification from Implementation:** Formalize domain goals and constraints prior to code generation.
2. **Enforce MDP Structure:** Force sequential decisions across parameter ranges, states, actions, and rewards.
3. **Constrain to One Design Question:** Focus each step via specific skill markdown files and checklists.
4. **Assist Expert with Domain Knowledge:** Use the "Trillian" persona to suggest answers under full human control.
5. **Carry Forward Decisions, Not Transcripts:** Pass compact step summaries to avoid context overload.
6. **Enable Iterative Reflection:** Use post-training diagnosis loops (Step 10) to refine MDP parameters.

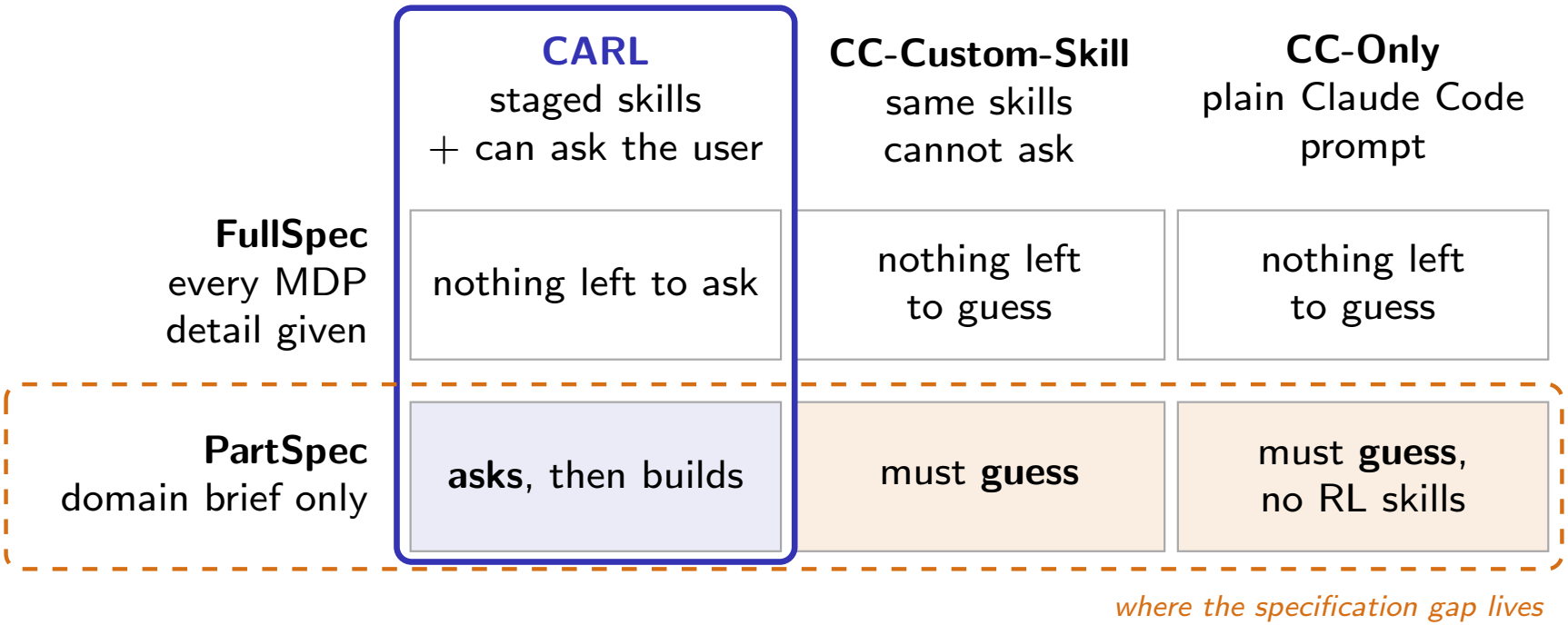
System Architecture & Technology Stack

- **Lightweight Python Orchestration Layer**
  - Replaces monolithic platforms with modular components (routes, conversation engine, session layer).
  - Supports a Flask web app and terminal client sharing a directory-backed session model.
  - Asynchronous background threads handle intensive model loops and training tasks.
- **Multi-Model Role Assignment**

Model Role	Assigned Engine	Thinking Setting
Main Workflow	Claude Opus 4.7	Variable (Med $\rightarrow$ Highest)
Trillian Suggestion	Claude Haiku 4.5	Narrow / Low-cost
Report & Editorial	Claude Sonnet 4.6	Parallel structured prose

- **Model-Agnostic Conversational Orchestration**
  - Interfaced via Bedrock Converse API, but logic remains completely uncoupled from any single vendor.
  - CARL controls step boundaries, context policies, tools, and state summaries.

Same agent, same task. Only the conversation differs.



The same frontier model (Claude Opus 4.7) writes the code in every cell.
Tested on two problems: **goals-based wealth management** & **Goddard rocket**.

What the user actually typed

FullSpec brief (excerpt)

- Binary 0/1 take-or-skip; investment action in $[0, 1]$, continuous
- 4-goal case: goal amounts in $[25, 125]$, horizons in $[10, 20]$ years
- Base utilities are integers in $\{1, \ldots, 8\}$
- **Final goals must be taken greedily**
- Parameters may be daily: **carefully simulate the evolution over 1 year**

PartSpec brief (excerpt)

- Choose whether to take the goal or skip it; allocation between all bond and all stock
- 1 to 4 goals, horizons of 10 to 20 years
- Goal amounts **such that the problems are reasonably feasible**
- *(nothing on unaffordable goals or the final year)*
- **A time step of 1 year for decisions.** *(daily-fitted parameters not mentioned)*

Both briefs: the conditional bonus **full text in paper, Appendix A**

Each goal already taken adds 25% to the utility of every later goal. Early, small goals act as multipliers for large later ones.

Three questions CARL asked before writing any code

The bonus rule — **Step 1**

CARL asked

Do prior-goal bonuses compound (1.25^k) or add ($1 + 0.25k$)?

Expert said

They add: each prior goal adds 25% of the *base* utility.

Became

Reward $\tilde{U}_i = U_i(1 + 0.25\, k_i)$, with k_i tracked in the state (Steps 4, 6).

Affordability — **Step 1**

CARL asked

Can a goal be taken without enough wealth? Is there an allocation decision at year T?

Expert said

No, and no: year T is a settlement date, not another investment period.

Became

Unaffordable "take" actions are masked; no allocation at T (Step 5).

Data frequency — **Step 1**

CARL asked

Parameters are fitted on daily prices. Simulate 252 sub-steps per year?

Expert said

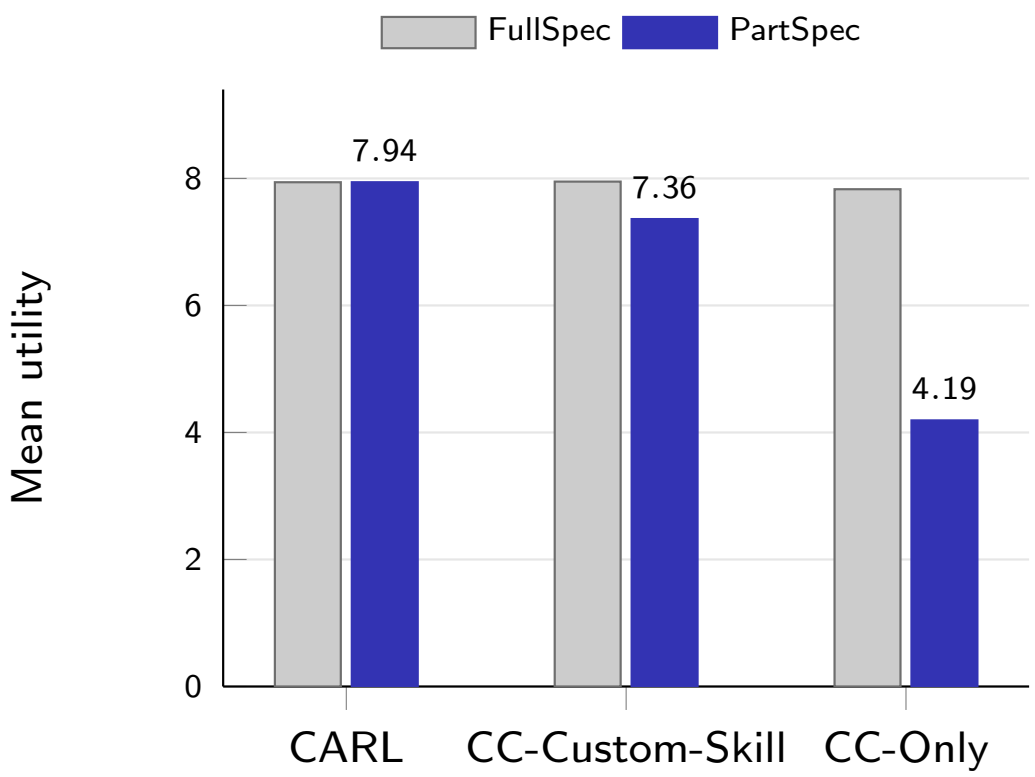
No: convert to annual-equivalent dynamics.

Became

Closed-form one-year GBM and Vasicek transitions with correlated shocks.

These aren't RL questions. **But they decide the MDP for RL.**

With a vague brief, CARL still matches the full spec



Increasing four-goal scenario, base utilities $(1, 1, 2, 4)$.

FullSpec bars: 7.94, 7.95, 7.83.

Each bar: 1,000,000 Monte Carlo episodes on identical market paths.

Agents that can't ask fall behind

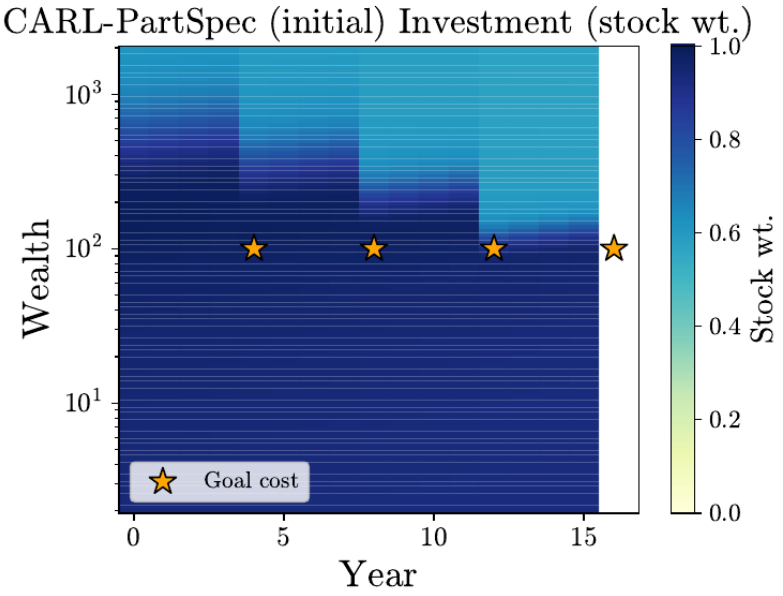
CC-Only spends greedily: takes goal 1 in 86% of episodes, then funds later goals only 32–44% of the time.

CC-Custom-Skill is too cautious: takes goal 1 in only 3% of episodes and forfeits the bonus multiplier.

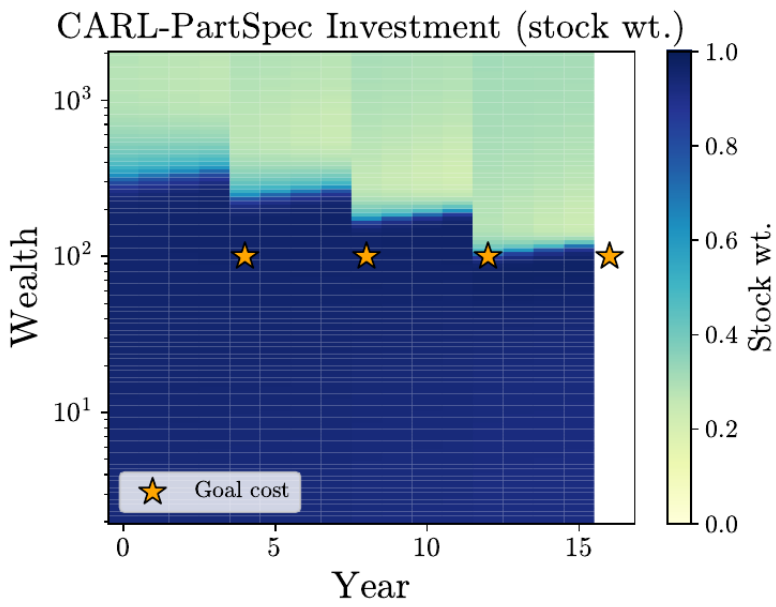
CARL-PartSpec takes goal 1 in 21% of episodes, the same rate as its full-spec twin (20%).

The expert saw what the score didn't

After the first run mean utility 7.80



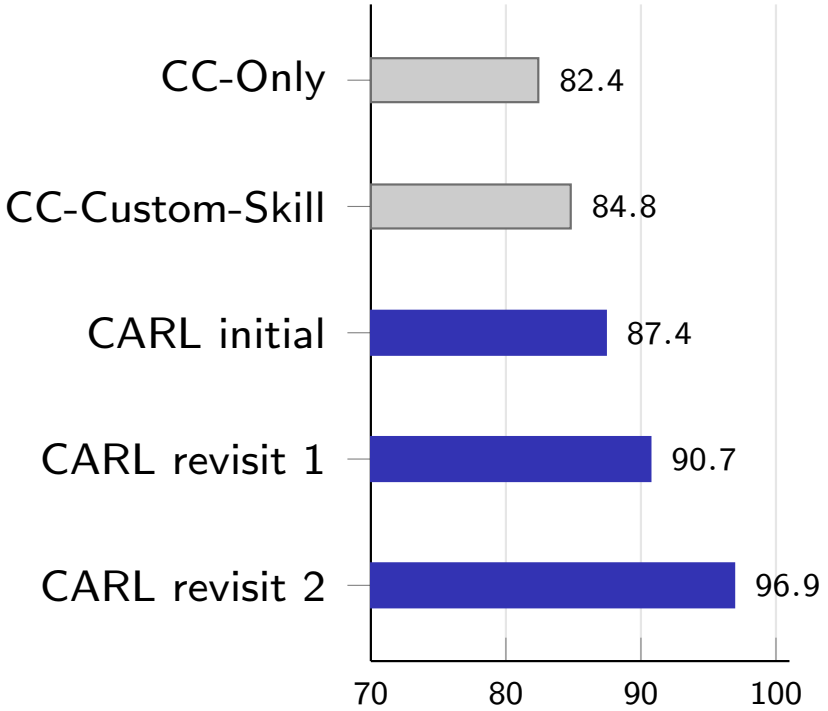
After the revisit (Step 10) mean utility 7.94



Expert flagged	CARL diagnosed	Fix
Still stock-heavy at high wealth; the stock-to-bond switch is blurry.	At high wealth the goals get funded anyway, so the objective is flat and the policy drifts toward its prior.	Lower Beta floor, plus an annealed terminal-wealth-variance penalty.

The score moved only slightly. The policy became the one a wealth manager would recognize.

Rocket Problem: Each conversation closed more of the gap



% of best, mean of 18 scenarios

Revisit 1 **Step 10**

CARL noticed: the rocket burns its fuel far earlier than the best heuristic.
Expert: thrust pays more later, in thinner air.
Became: a one-sided early-burnout penalty, plus two state features that make it predictable.

Revisit 2 **Step 10**

Expert: fit the target burnout time to the heuristic's actual burnouts.
Became: $t_{\text{target}} = 75x - 8x^2$, with $x =$ fuel$/1000$.

Best per scenario is Unconstrained Optimal or MC-Local-Greedy. CARL: under 1 s per rollout; MC-Local-Greedy: 20–234 s. Paper, Tables 6–7; full tables in the appendix.

Core Architectural Conclusions

1. **The Value of Staged Dialogue:** Conversational scaffolding converts incomplete briefs into well-posed MDPs without requiring complete specifications upfront.

   → the bonus-rule question; CARL-PartSpec matches CARL-FullSpec (7.94 vs. 7.94)

2. **Specification Quality vs. Coding Speed:** Poor RL performance is rarely caused by hyperparameters or basic coding errors; it stems from structural oversights in the MDP design itself.

   → same model, same skills, only the brief differs: CC-Custom-Skill falls from 7.95 to 7.36, CC-Only from 7.83 to 4.19

3. **Iterative Pipeline Reality:** Rather than expecting perfect specification upfront, complex engineering requires tools that can capture adjustments during the realization phase.

   → the revisit: the expert flagged a blurry stock-to-bond switch; the fix lifted utility from 7.80 to 7.94

Extensions and Future Work

Broader RL Domains & User Base

- **Complex RL Domains:** Extend CARL beyond compact environments to partially observable, multi-agent, logistics, robotics, and healthcare systems.
- **Usability & Interaction Studies:** Evaluate workflow friction, clarification turn counts, and user trust across varying levels of RL expertise.

Adaptive Interaction Policies

- **Beyond Single Persona:** Dynamically adjust interaction depth, confidence thresholds, and query frequency based on user profile and domain complexity.

Responsible AI & Auditability

- **Out-of-Sample Stress Testing:** Develop systematic evaluation protocols for non-analytical benchmark problems.
- **Decision Trace Auditing:** Maintain clear intent-to-MDP audit trails and policy explainability for regulated sectors (e.g., finance, healthcare).

CARL as a General Design Pattern

- **Meta-Workflow Concept:** Generalize the staged specification paradigm to build specialized assistants for scientific computing and simulation design.

Summary

"AI is overhyped in the short term and drastically underestimated in the long term." – Demis Hassabis

We discussed the following topics:

- Agentic AI – Escape Capability overhang
- RL for Retirement planning
- AI Agents to solve the RL Specification Gap and democratize RL solutions

THANK YOU!