From Reasoning Models to Agents

Lecture 6

Elodie Chervin Daniel Barbosa

TT 2026

University of Oxford & Nuffield College

Roadmap

From next-token prediction to reasoning

From reasoning to agents

Coding-agent components

From next-token prediction to reasoning

Start with the base object

A language model represents a distribution over token sequences:

$$P(w_1, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_{<t}).$$

This is powerful, but it is still a token-in, token-out object.

The rest of the lecture is about changing the training target and the environment around that object.

LLM, reasoning model, agent

Conventional LLM

Reasoning LLM
(more expensive to run)

LLM in agent harness

Three increasingly capable systems

LLM

- Predicts and completes text
- Follows instructions after post-training

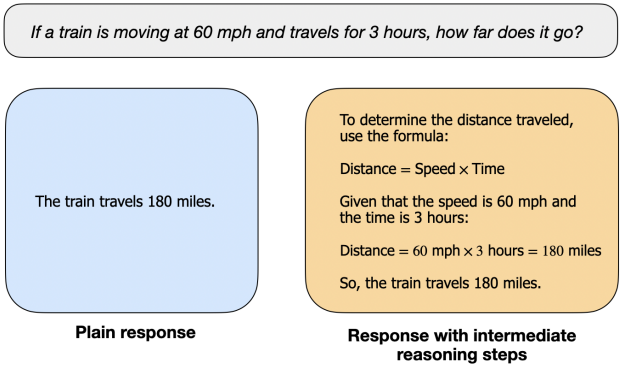
Reasoning model

- Spends more inference compute
- Searches, verifies, revises

Agent

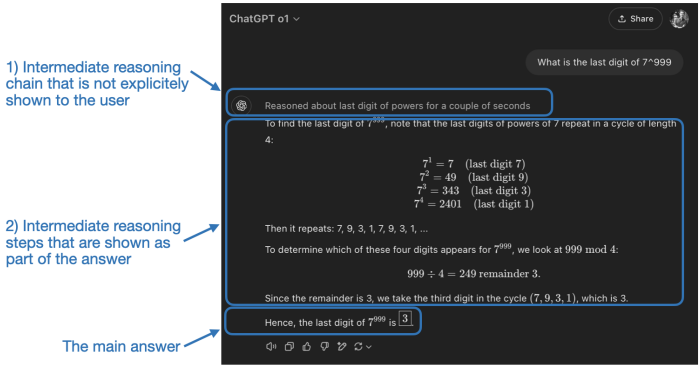
- Observes an environment
- Calls tools and updates state

What does a reasoning model add?



A reasoning model is useful when the answer depends on intermediate steps, not just recall.

Reasoning can happen in two places



- **Internal**: the model spends hidden compute before producing the final answer.
- **External**: the model writes intermediate steps for the user or tool harness to inspect.

Prompting for reasoning

Weak prompt

```
Solve this regression problem.
```

Better prompt

```
Break the problem into steps.
State the estimand, identify the
data transformation, then check
units before giving the answer.
```

For coding

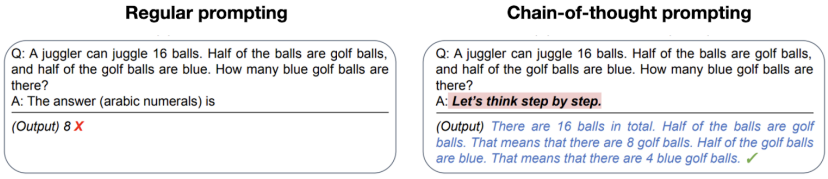
```
Before editing, inspect the failing
test, name the likely cause, make
one change, then rerun the test.
```

For data work

```
Compute the statistic twice:
first directly, then with a
sanity-check aggregation.
```

Prompting can buy reasoning at inference time; it does not change the model weights.

Chain-of-thought is an inference-time trick



Asking for intermediate steps increases token use, but often improves hard multi-step tasks.

When is a reasoning model worth it?

Good at	Bad at
+ Deductive or inductive reasoning (e.g., riddles, math proofs)	– Fast and cheap responses (more inference time)
+ Chain-of-thought reasoning (breaking down multi-step problems)	– Knowledge-based tasks (hallucination)
+ Complex decision-making tasks	– Simple tasks (“overthinking”)
+ Better generalization to novel problems	

Knowledge-based tasks: tasks mainly answered from known facts, definitions, or supplied context; little search, calculation, or verification is needed.

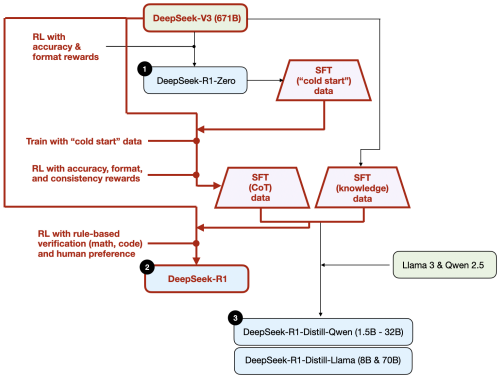
Use the expensive model when the task has structure worth searching through.

Fine-tuning: changing the model's habits

Prompting	Put instructions in the input. The model can comply, but its weights are unchanged.
Supervised fine-tuning (SFT)	Train on examples of good inputs and outputs: solutions, traces, explanations, code patches.
Distillation	Use a stronger reasoning model to generate training examples for a smaller model.

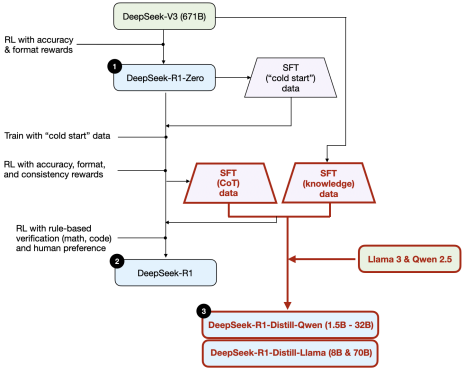
Fine-tuning teaches a pattern; RL then rewards which learned behaviours actually work.

From SFT to RL



SFT gives the model examples of good reasoning traces; RL rewards correct, checkable outcomes.

Distillation: making reasoning cheaper



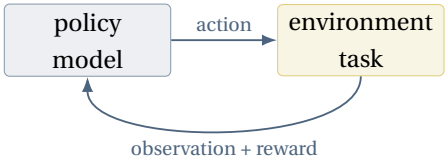
Smaller models can learn from traces generated by stronger models, even when they cannot discover the behaviour from scratch.

Four levers for building reasoning models

Lever	What it changes
Inference-time scaling	Spend more compute at answer time: prompting, sampling, voting, search.
SFT	Show the model what good reasoning traces and final answers look like.
RL	Reward verifiable behaviour directly: tests pass, math checks, required format.
Distillation	Transfer reasoning behaviour from a stronger model into a cheaper model.

Reasoning improves when we add search, examples, feedback, or transfer.

Reinforcement learning (RL) in one slide



- **Policy**: the current model; it chooses an answer, reasoning trace, or tool action.
- **Environment**: the task plus checker: unit tests, grader, human preference, or reward model.
- **Reward**: a score that says how good the action was, e.g. test passes, answer preferred, proof checks.
- **Update**: make rewarded behaviour more likely next time; make failed behaviour less likely.

For LLMs, RL turns feedback into pressure on the model's output distribution.

Why RL helps reasoning models

- Some tasks have verifiable answers: tests pass, proofs check, calculations balance.
- Outcome rewards can push models toward correct final answers.
- Process rewards can push models toward better intermediate work.
- More inference-time compute lets the model plan, test, and revise before answering.

Reasoning is useful when the task has structure that can be checked.

But reasoning is not a guarantee

- Extra reasoning tokens increase cost and latency.
- A fluent chain can still rationalize a wrong answer.
- Hidden reasoning is not an audit trail.
- External checks matter: code, citations, tests, units, dates.

For research use, prefer auditable summaries plus tool-verified evidence.

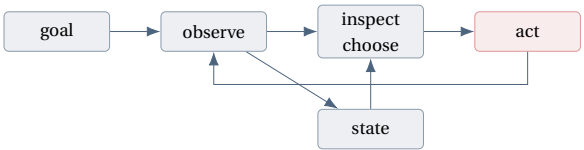
From reasoning to agents

An agent is a control loop around a model.

It repeatedly decides:

1. what to inspect,
2. which tool to call,
3. how to update state,
4. whether to continue or stop.

The loop



Economics example

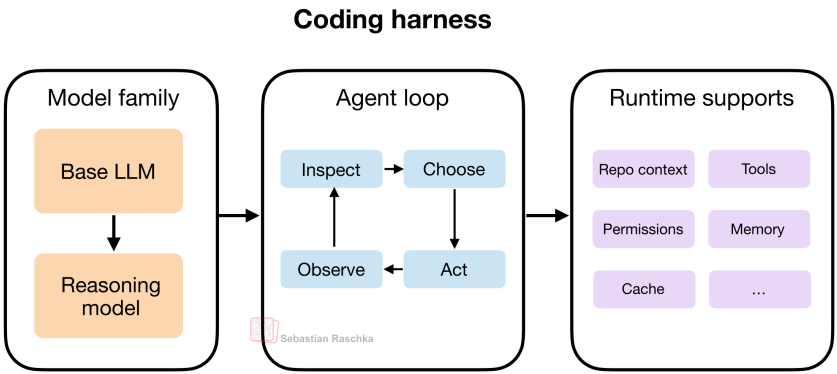
- Goal: replicate a CPI inflation chart.
- Observe: read README, data folder, and notebook errors.
- Inspect/choose: decide whether to fetch FRED data or fix code.
- Act: run Python, edit the script, write the memo.
- State: files read, commands run, failures, and decisions.

Loop: observe, inspect/choose, act; state grounds the next step.

Why coding agents feel different from chat

- Chat suggests commands; an agent can run commands.
- Chat describes files; an agent can read and edit files.
- Chat guesses test failures; an agent can inspect logs.
- Chat forgets context; an agent can persist memory.

A coding harness has three layers

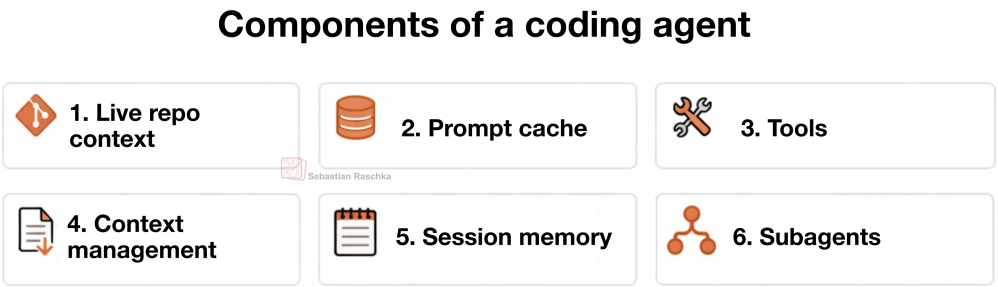


Checkpoint: chatbot or agent?

System behaviour	Classification
Summarizes an uploaded PDF once	Chat with context
Searches a repo, edits a file, runs tests, reads the error, edits again	Agent
Produces a detailed plan but asks the user to execute every step	Chat assistant
Retrieves CPI data, computes inflation in Python, writes a memo with caveats	Agentic research workflow

Coding-agent components

Six components

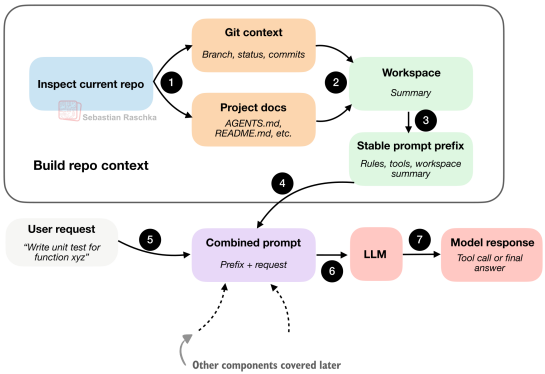


1. Live repo context

- **Repository root and layout**: where code, data, notebooks, and outputs live.
- **Version control**: branch, diff, and working tree status.
- **Instructions**: README, AGENTS.md, assignment notes, or project conventions.
- **Tests**: commands that check the work, e.g. `pytest`, notebook execution, data validation.
- **Package files**: dependency and entry-point files such as `pyproject.toml`, `requirements.txt`, or `package.json`.
- Relevant files and symbols for the current task.

A request like “fix the tests” is meaningless without workspace facts.

Workspace summary

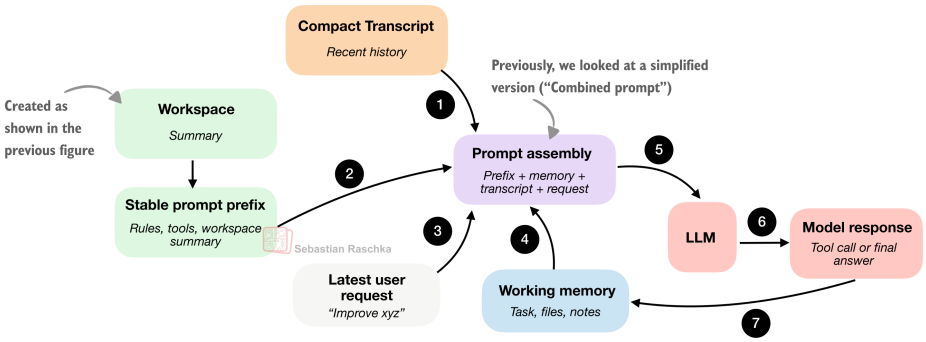


2. Prompt shape and cache reuse

$$\text{model input}_t = \text{stable prefix} + \text{compact state}_t + \text{new request}_t.$$

The stable prefix contains rules, tool schemas, and workspace facts; the compact state changes each turn.

Prompt packaging



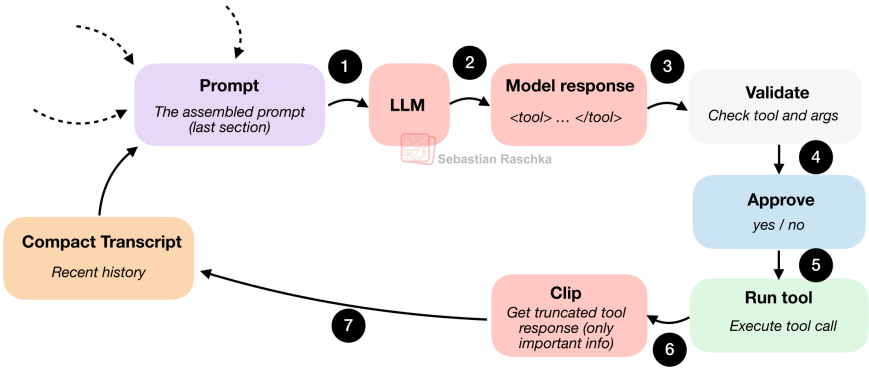
3. Tools, validation, permissions

Tool: a typed action the harness can execute outside the model, such as reading a file, running Python, querying an API, or editing code.

- Named and typed: `fetch_fred_series(series_id, start, end)`.
- Validated: series exists; dates are valid; output has expected columns.
- Permissioned: ask before overwriting data, sending email, or pushing code.
- Bounded: return a summary plus first/last rows, not a 50,000-line table.

Economics example: fetch CPI, compute year-on-year inflation in Python, then write a memo with caveats.

Tool-use flow

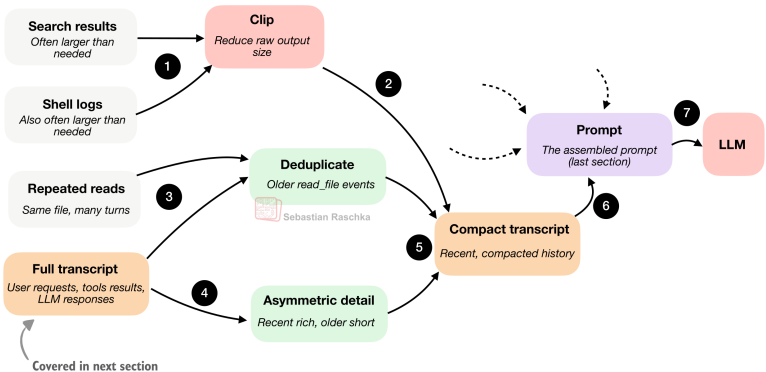


4. Context reduction

- **Context**: the information inside the model input right now: request, rules, files, tool outputs, summaries, and recent decisions.
- Logs, files, and repeated tool outputs quickly bloat the context.
- Older events should usually be compressed more aggressively.
- Repeated file reads should be deduplicated.
- Recent failures and user decisions should stay rich.

Context quality often looks like model quality.

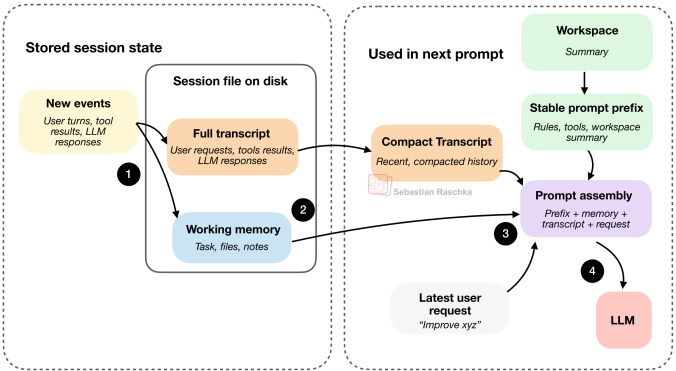
Compaction


Search results
Often larger than needed
Shell logs
Also often larger than needed
Repeated reads
Same file, many turns
Full transcript
User requests, tools results, LLM responses
Clip
Reduce raw output size
Deduplicate
Older read_file events
Asymmetric detail
Recent rich, older short
Compact transcript
Recent, compacted history
Prompt
The assembled prompt (last section)
LLM
1
2
3
4
5
6
7
Sebastian Raschka
Covered in next section

5. Session memory

Memory layer	Job
Full transcript	Durable record of requests, tool calls, outputs, and responses.
Working memory	Short list of active facts, files, decisions, and blockers.
Prompt summary	Compressed context reconstructed for the next model call.

Session state



6. Delegation

Subagents help when a side task can be bounded.

Good side tasks

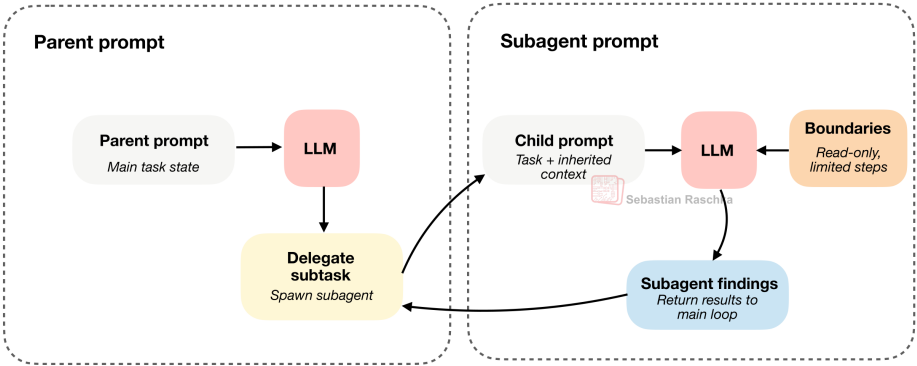
- Read-only search for a symbol.
- Inspect one failing test.
- Summarize a configuration file.
- Compare two implementation options.

Economics research example

- Ask one subagent to audit CPI units and seasonal adjustment.
- Ask another to inspect the replication script failures.
- Keep writing, interpretation, and final claims with the main agent.

Delegate bounded checks; keep synthesis and judgement centralized.

Bounded subagents



Takeaways

1. A reasoning model is still an LLM, but trained and used to spend more compute on hard tasks.
2. An agent is a loop around a model, not magic inside the model.
3. Coding agents work well because repos provide readable state and executable feedback.
4. Harnesses matter because tools, context, memory, and permissions shape the work surface.

Further reading and watching

- Raschka, *Understanding Reasoning LLMs*
  magazine.sebastianraschka.com/p/understanding-reasoning-llms
- Raschka, *Components of a Coding Agent*
  magazine.sebastianraschka.com/p/components-of-a-coding-agent
- Korinek, *AI Agents for Economic Research*
  nber.org/papers/w34202
- Panjwani, *OpenAI Codex Full Course 4 Hours: Build & Ship*
  youtube.com/watch?v=j7d5rs0iMlE
- Panjwani, *AI Agents for Economics Research* (VoxDev talk)
  voxdev.org/event/ai-agents-economics-research

Questions?

Models generate tokens.

Harnesses turn tokens into work.