WHAT HAPPENS BETWEEN THE GOAL AND THE ACTION?

Meet Agent Reasoning Patterns — reusable control loops for deciding, acting, checking and improving.

ONE GOAL → DIFFERENT CONTROL LOOPS

DIRECT	Answer / act
ReAct	Decide ↔ act ↔ observe
REFLECT	Try → feedback → improve
SEARCH	Explore alternatives

Reasoning patterns are ways to control the agent loop — not one universal recipe.

REASONING ≠ PLANNING ≠ ACTING

They overlap, but each solves a different part of the agent loop.

REASONING

Choose what makes sense next.

PLANNING

Organize steps toward the goal.

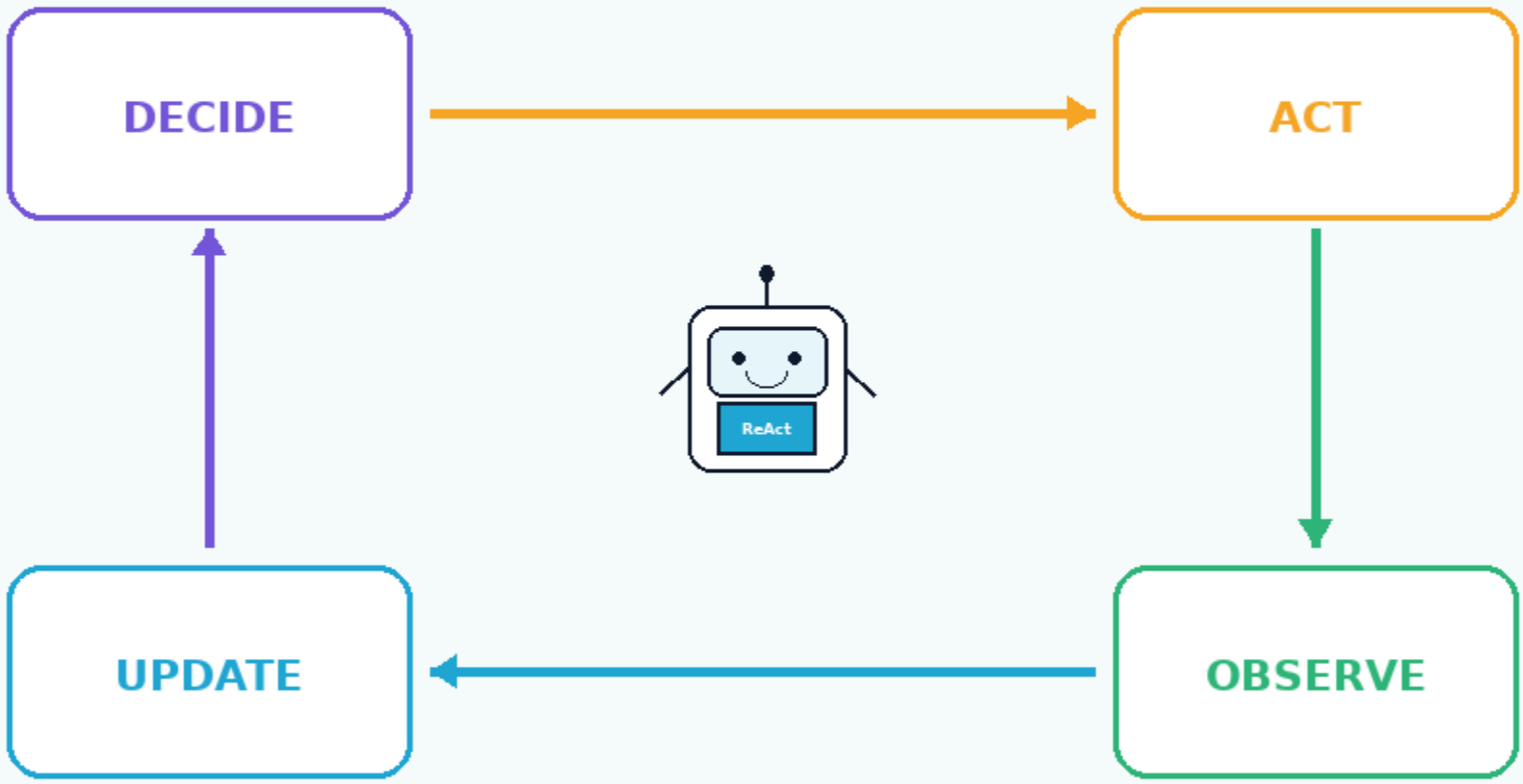
ACTING

Use tools or change the world.

A strong agent can reason locally, plan globally and act through tools.

ReAct: DECIDE → ACT → OBSERVE → REPEAT

ReAct interleaves decision-making with actions and fresh observations.



Each observation can change the next action instead of relying on one fixed guess.

REAL ReAct EXAMPLE: RESEARCH AN AI STARTUP

The agent gathers evidence step by step instead of fabricating the whole answer.

1 **Need current funding data**

2 **Search web**

3 **Observe: latest round found**

4 **Need founder background**

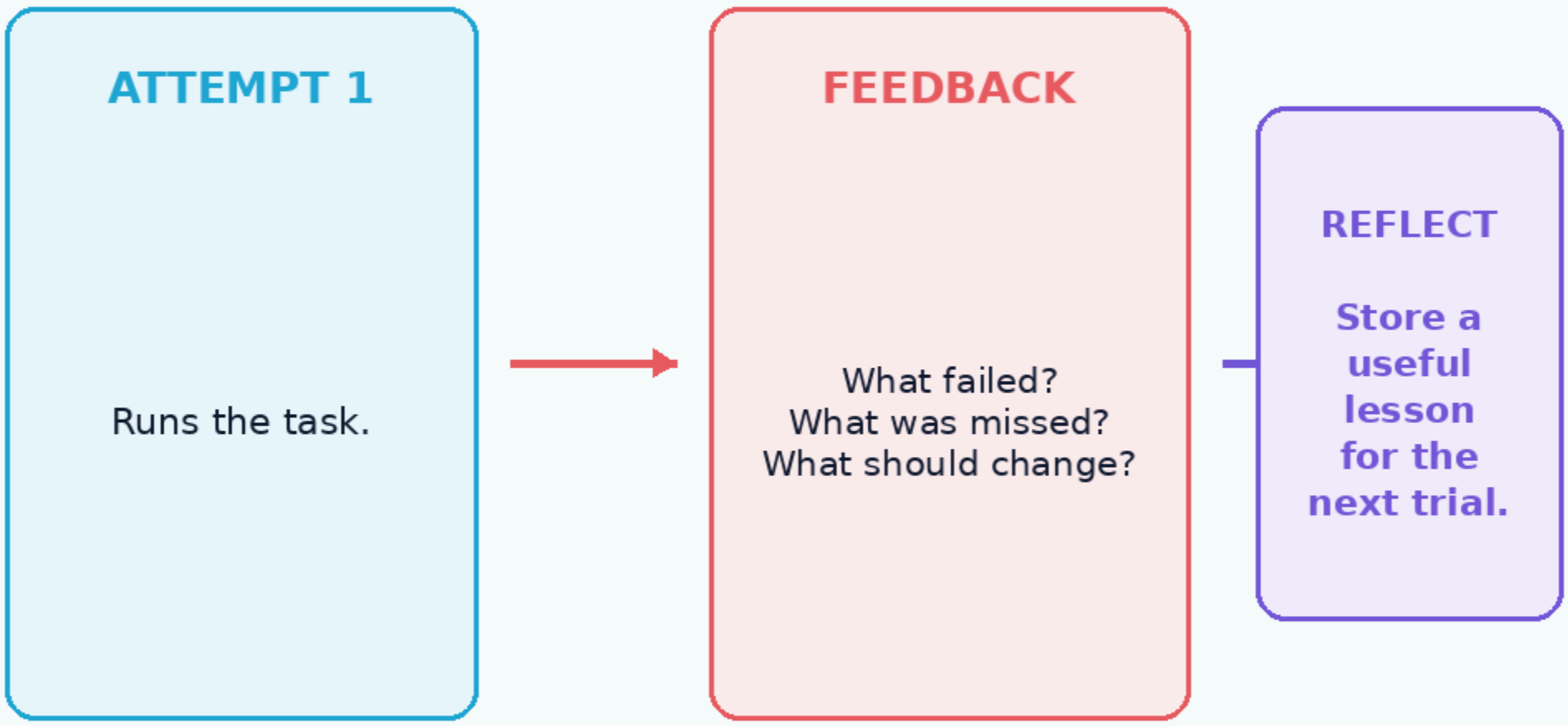
5 **Search another source**

6 **Synthesize supported answer**

ReAct is especially useful when external information changes what the agent should do next.

REFLECTION: LEARN FROM THE LAST ATTEMPT

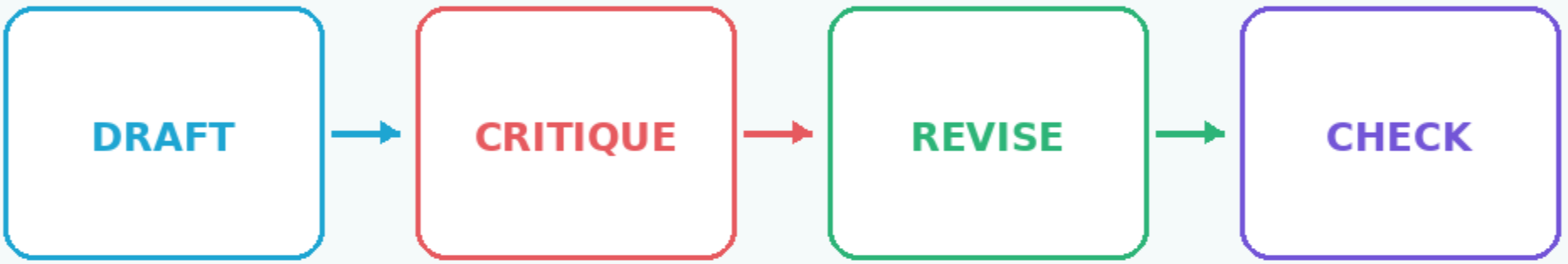
Reflection uses feedback from a completed or failed attempt to improve the next one.



Reflection is most useful when it is anchored to real feedback, not vague self-critique.

SELF-REFINE: DRAFT → CRITIQUE → REVISE

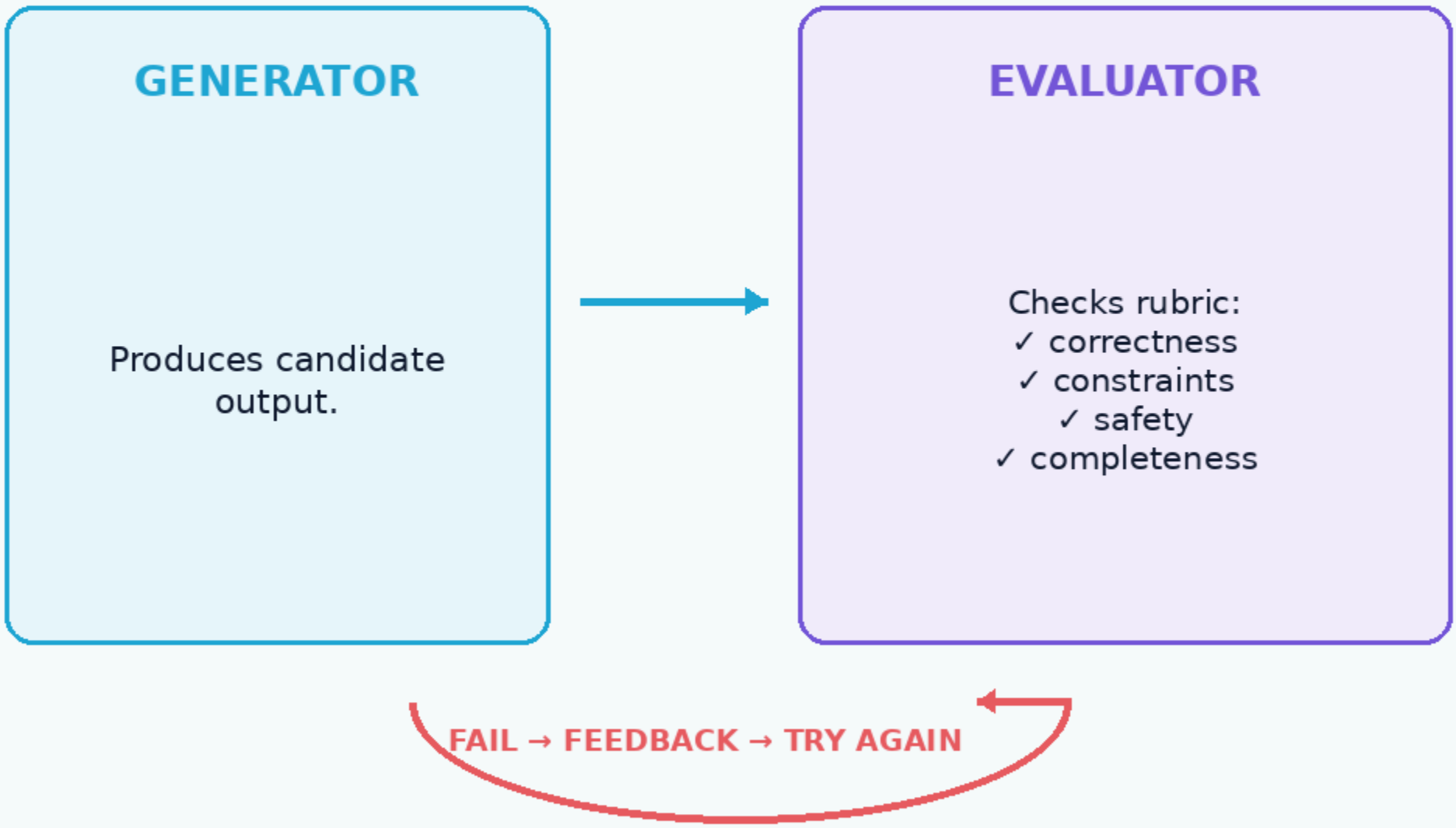
One output can be iteratively improved before it is returned.



Refinement is a loop around one candidate — not necessarily a search over many candidates.

EVALUATOR–OPTIMIZER: SEPARATE GENERATION FROM JUDGMENT

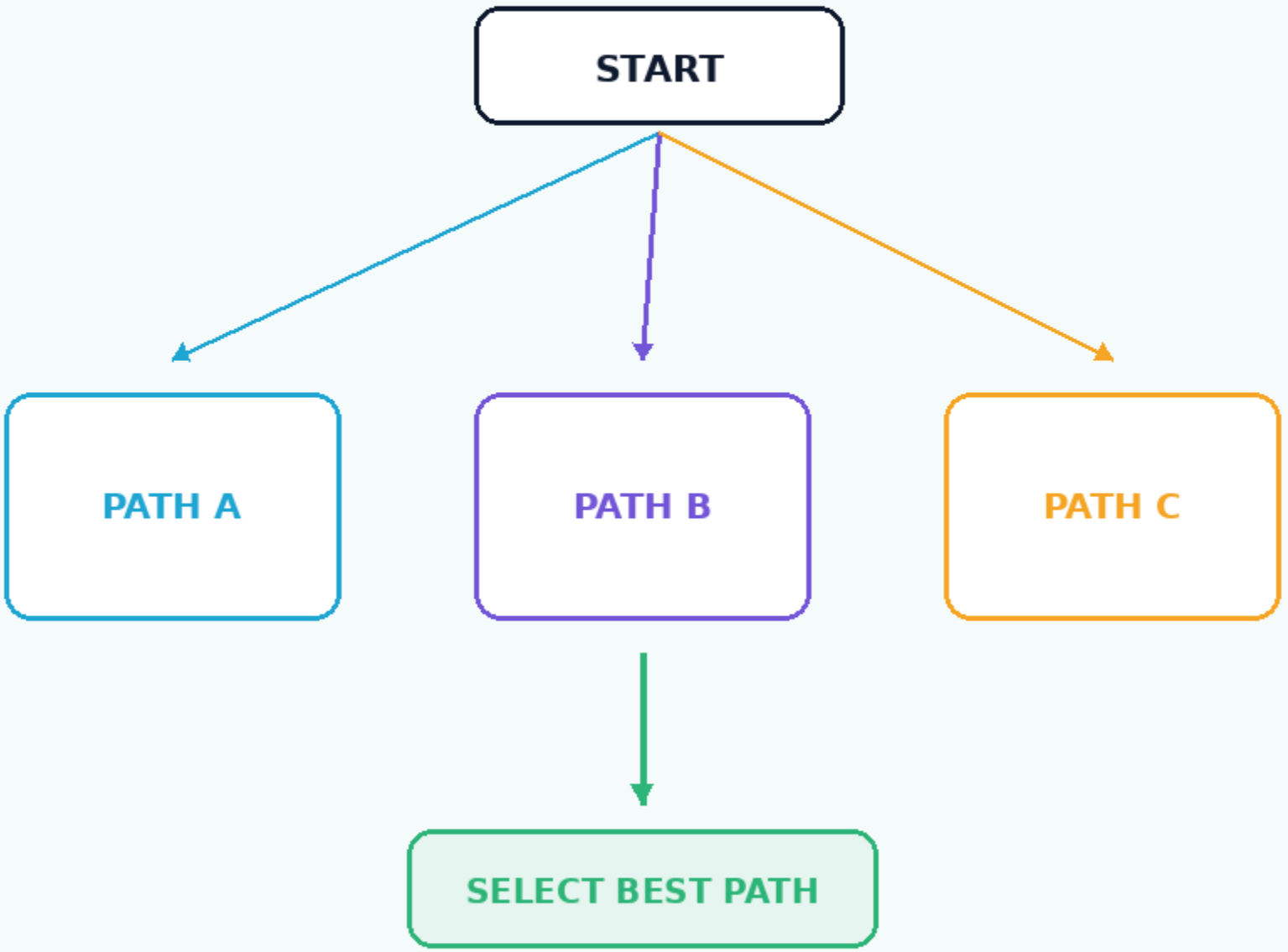
A generator proposes. An evaluator scores against explicit criteria.



Separating generation and evaluation can make quality criteria more explicit.

WHEN ONE PATH IS NOT ENOUGH: SEARCH

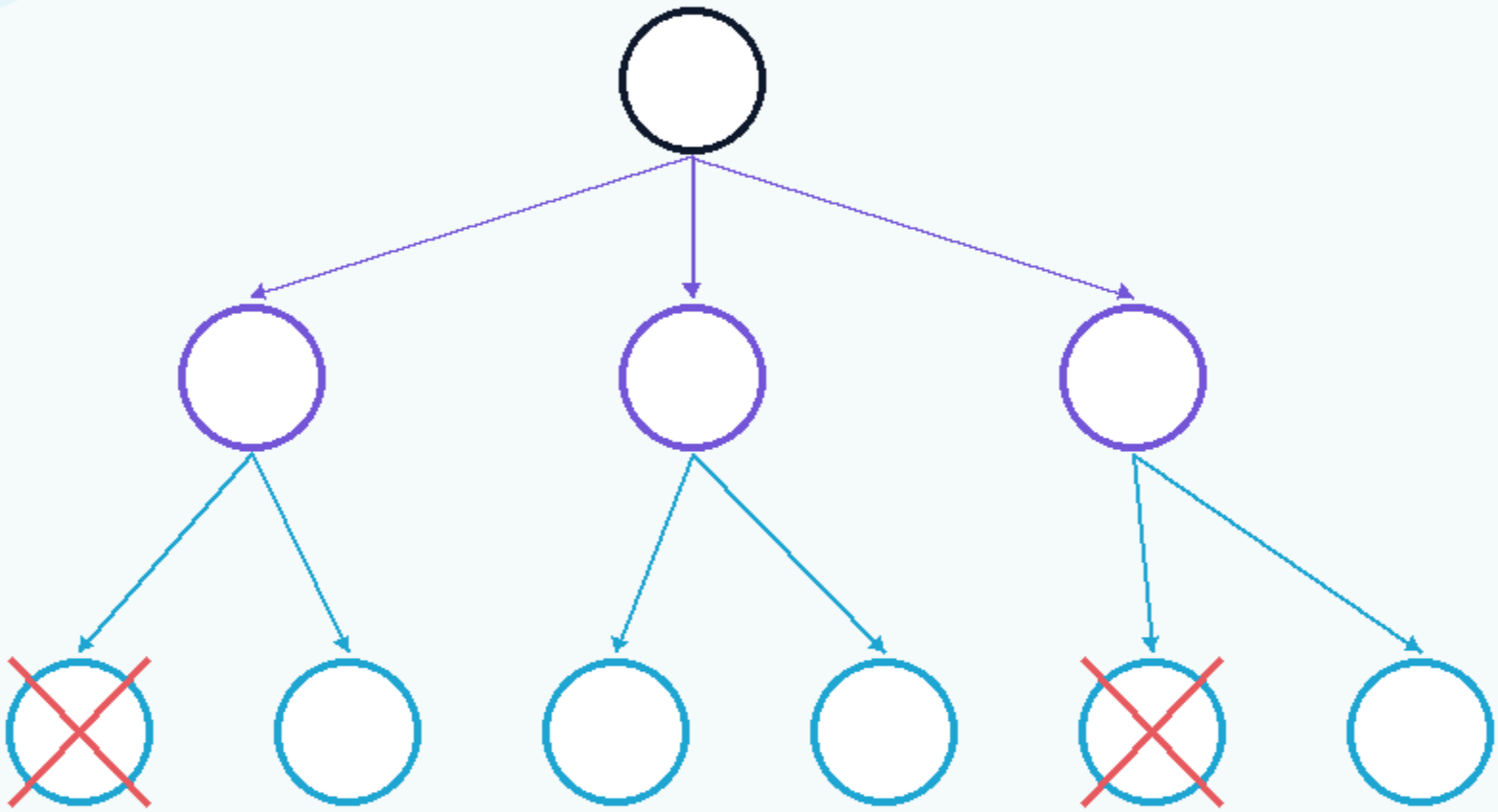
Some problems benefit from exploring alternatives before committing.



Search spends more compute to reduce the risk of committing to a poor early choice.

TREE OF THOUGHTS: BRANCH, SCORE, CONTINUE

Tree-style search explores multiple intermediate possibilities and can backtrack.



SCORE → PRUNE → EXPAND

Tree search is useful when exploration and backtracking matter more than fast one-pass execution.

SEARCH NEEDS A SCORING FUNCTION

Branching without evaluation only creates more possibilities.

PROMISE	How likely is this path to succeed?
CONSTRAINTS	Does it satisfy hard requirements?
COST	How expensive is the next expansion?
RISK	Could this path create harmful side effects?

A search strategy is only as useful as the signal used to rank its alternatives.

WHICH PATTERN SHOULD YOU USE?

Choose by task complexity, uncertainty and feedback availability.

DIRECT	Simple + obvious next step
ReAct	Need tools and fresh observations
REFINE	One candidate can improve iteratively
REFLECT	Learn from a failed/completed trial
SEARCH	Early choices are uncertain or strategic

More sophisticated reasoning is not automatically better — it must earn its extra cost.

EXTERNAL FEEDBACK BEATS PURE SELF-TALK

Ground loops in signals the system can actually verify.

TESTS	Did the code pass?
TOOL RESULTS	What did the API return?
ENVIRONMENT	Did the action change the world?
HUMAN	Was this approved?
EVALUATOR	Did the output satisfy the rubric?

Feedback loops become much stronger when they can observe something outside the model.

STOP CONDITIONS ARE PART OF REASONING

Without stopping rules, a useful loop can become an expensive loop.

CONTINUE WHEN

New evidence can improve the result.

A required step remains.

The evaluator gives actionable feedback.

STOP WHEN

Goal is satisfied.

No useful progress.

Budget / step limit reached.

Policy blocks continuation.

A reasoning loop needs both a next-step rule and a stop rule.

OBSERVABILITY DOES NOT REQUIRE PRIVATE CHAIN-OF-THOUGHT

Production systems can stay inspectable by logging actions, state and outcomes.

LOG THIS

Tool calls
Arguments
Results
State transitions
Scores
Errors
Approvals

DO NOT RELY ON

Raw hidden reasoning as the only debugging signal.

Inspect behavior through traces and structured evidence — not by depending on private reasoning text.

BETTER REASONING COSTS SOMETHING

Every extra branch, critique or evaluator call adds latency and tokens.

QUALITY

Can improve on hard tasks

LATENCY

More model/tool rounds

COST

More tokens + compute

COMPLEXITY

More failure modes

Use deeper loops only where the expected quality gain justifies the extra work.

REAL EXAMPLE: A CODING AGENT

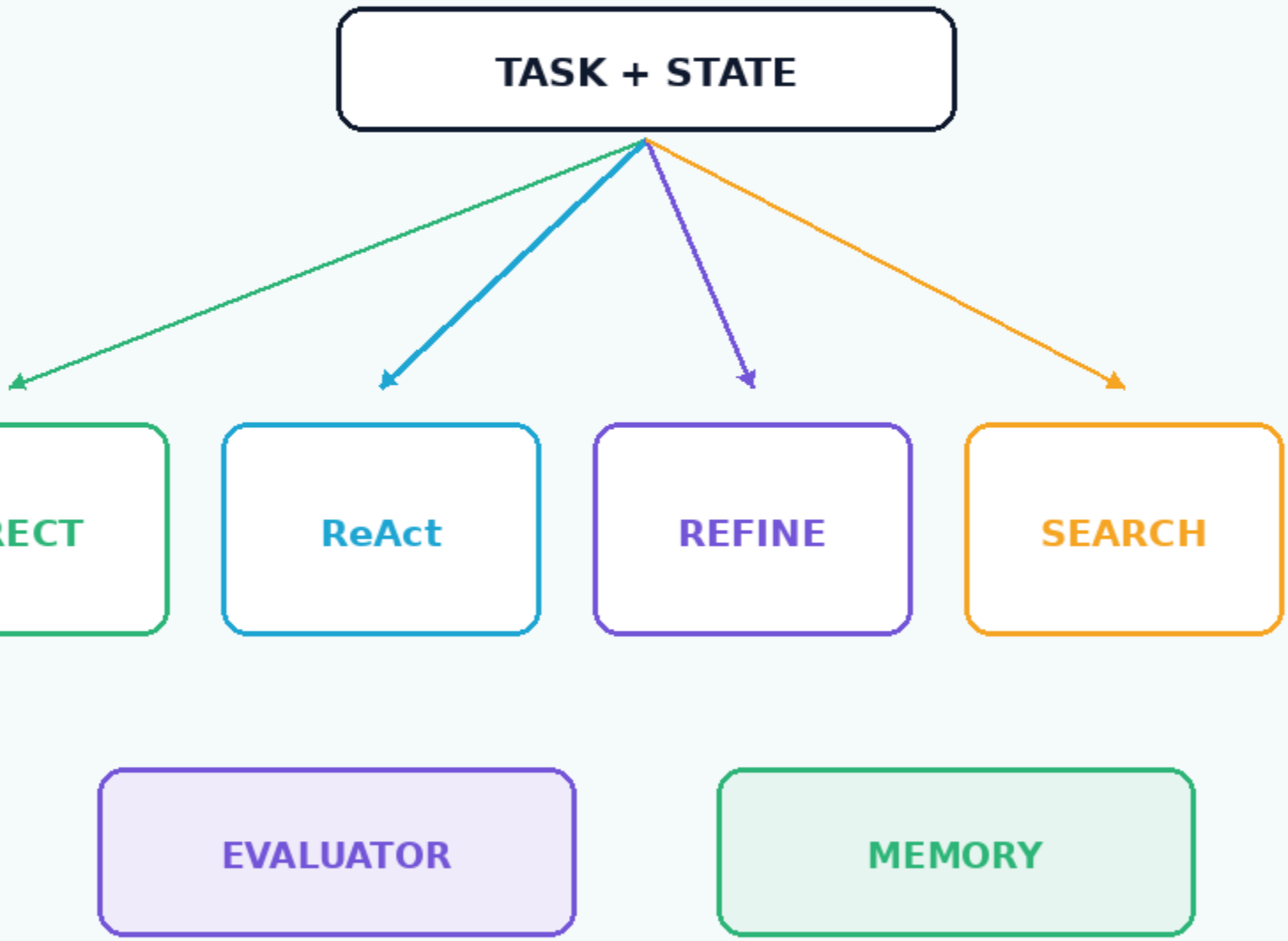
Different reasoning patterns can appear in one task.

1. Read task + repo
2. Plan likely files
3. Edit code
4. Run tests → observe failure
5. ReAct: inspect error + edit
6. Evaluator: run full regression
7. Reflect: save lesson if useful

Real agents often combine patterns rather than using one pattern for the entire run.

A PRACTICAL REASONING STACK

The harness can choose different loops based on the task.



Reasoning is a runtime choice inside the harness — not a single hard-coded prompt trick.

DAY 24 TAKEAWAY

Reasoning patterns are control loops for deciding, checking, exploring and improving.

DIRECT | ReAct | REFINE | REFLECT | SEARCH

ACT ON EVIDENCE

Use observations to choose next steps

EVALUATE

Check whether the candidate is actually good

EXPLORE

Branch only when alternatives matter

STOP

End loops when more work adds little value

THE GOAL IS NOT “MORE THINKING.”
THE GOAL IS BETTER DECISIONS PER UNIT OF COST.

NEXT: DAY 25 — Agent Security: prompt injection, permissions, tool risk and data boundaries.