Prompt Engineering / Reasoning Systems

Start from the desired conclusion and work backward to find what premises would support it.
Difficulty: Advanced
Model: GPT-4 / Claude / Gemini
Use Case: Goal-Directed Planning, Root Cause Analysis, Proof Construction
Updated: May 2026
Why This Prompt Exists
Forward reasoning starts from what you know and asks “what follows?” Backward reasoning starts from where you want to be and asks “what would get me there?”

You get:

- forward reasoning that gets lost in irrelevant details
- no way to work toward a specific goal efficiently
- solutions that are correct but not optimal for your goal
- difficulty solving problems where the goal is clear but the path isn’t
- wasted exploration of states that don’t lead to the goal

But backward reasoning is efficient:

- goal state: what we want to achieve
- backward step: what must be true just before the goal?
- recursive: repeat until reaching initial state
- branching: multiple ways to achieve the same subgoal
- pruning: discard subgoals that can’t be reached

Without backward reasoning, you wander aimlessly.

This prompt implements backward (goal-driven) reasoning.

The Prompt

```
Assume the role of a backward reasoning engine that works from goals
to premises.

Your task is to start from the desired conclusion and work backward to
find supporting premises.

Generate:

1. GOAL STATE (what we want to prove/achieve)
   - Clear statement of the goal

2. BACKWARD STEP 1
   - What conditions would directly imply the goal?
   - List 2-3 possible immediate predecessors
   - For each: state the condition and why it would lead to the goal

3. BACKWARD STEP 2 (for each promising predecessor)
   - What conditions would imply each predecessor?
   - Continue expanding backward

4. TERMINATION CONDITIONS
   - When do we stop? (Reached given premises / Contradiction / Max
depth)

5. COMPLETE BACKWARD CHAIN (chosen path)
```

   - Goal ← Condition A ← Condition B ← Given premises

6. FORWARD VERIFICATION
   - Starting from given premises, follow the chain forward
   - Does it reach the goal? (Yes/No)

7. ALTERNATIVE PATHS (if any)
   - Other backward chains that also reach the goal

INPUTS:

Goal statement:
[E.G., "Prove that the triangle is equilateral"]

Given premises/starting state:
[E.G., "We know sides AB = BC and angle B = 60 degrees"]

Available rules/inference steps:
[E.G., "Geometric theorems, algebraic operations"]

Max backward depth:
[3 / 5 / 10]

Model:
[GPT-4 / CLAUDE / GEMINI]

RULES:
- Backward reasoning is efficient when the goal is clear and the state space is large

```
- Branch conservatively (2-3 options per step) to avoid explosion
- Stop when you reach given premises (success) or contradiction
(failure)
- Always verify forward (backward reasoning can introduce false
dependencies)
- Use for: theorem proving, planning, root cause analysis, diagnosis
```

How To Use It

- Use when the goal is clear but the path is not (diagnosis, planning, proof construction).
- Branch conservatively — too many alternatives will explode combinatorially.
- Always verify forward after backward reasoning (backward may create false dependencies).
- Stop at contradiction early — no need to explore further.
- Backward reasoning is especially useful for root cause analysis (start with symptom, work backward to cause).

Example Input

Goal statement:
"Website conversion rate increases by 15%"

Given premises/starting state:
"Current conversion rate is 3%, we have $10,000 budget, 3 months timeline"

Available rules/inference steps:
"Marketing tactics (email, ads, SEO), their typical conversion lift, and cost"

Max backward depth:
"4"

Why It Works

Forward reasoning explores all possibilities from given premises — most of which don't lead to the goal.

This framework improves outcomes by forcing:

- goal specification (what are we trying to achieve?)
- backward step generation (what would get us there?)
- recursive expansion (work backward until we reach given premises)
- forward verification (check that backward chain actually works)
- alternative path discovery (other ways to reach the goal)

Great backward reasoning doesn't explore randomly — it explores only paths that could lead to the goal.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

See also Metacognition Scaffold