DeepSeek-R1

Deep Dive into DeepSeek — Chapter 9

Yin Yang

Foundation Books Project

The one rule of this chapter

R1-Zero showed that rule-reward RL can elicit reasoning.
DeepSeek-R1 solves the harder problem: making that behavior
readable, measurable, and usable as a release.

- Cold-start data shapes a readable actor before RL, not after.
- Reasoning-oriented RL, rejection sampling, mixed SFT, and final RL form a *branched* topology — not one straight chain.
- The reasoning-RL checkpoint *generates* training data; DeepSeek-V3-Base *initializes* the run that trains on it. These are two different edges, confused constantly in secondhand summaries.

Goal: follow one verifiable prompt through cold start, reasoning RL, rejection sampling, mixed SFT, final RL, distillation, and release evaluation — and see where the public recipe's boundary actually sits.

Roadmap

1. A release problem, carried by one prompt

Roadmap

1. A release problem, carried by one prompt
2. The pipeline as a branch of policies

Roadmap

1. A release problem, carried by one prompt
2. The pipeline as a branch of policies
3. Cold start, reasoning RL, and rejection sampling

Roadmap

1. A release problem, carried by one prompt
2. The pipeline as a branch of policies
3. Cold start, reasoning RL, and rejection sampling
4. Executing an R1-style loop with veRL

Roadmap

1. A release problem, carried by one prompt
2. The pipeline as a branch of policies
3. Cold start, reasoning RL, and rejection sampling
4. Executing an R1-style loop with veRL
5. Distillation and release evaluation

Roadmap

1. A release problem, carried by one prompt
2. The pipeline as a branch of policies
3. Cold start, reasoning RL, and rejection sampling
4. Executing an R1-style loop with veRL
5. Distillation and release evaluation
6. Synthesis: R1 versus R1-Zero

A release problem, carried by one prompt

Same mathematics, four stage witnesses

Find all positive integer pairs (a, b) *with* $a + b = 17$, $ab = 60$. *Return the pairs and a short verification.*

Stage witness	Schematic behavior
Base response	May name $5, 12$ but omit ordered pairs or verification

Same mathematics, four stage witnesses

Find all positive integer pairs (a, b) *with* $a + b = 17$, $ab = 60$. *Return the pairs and a short verification.*

Stage witness	Schematic behavior
Base response	May name $5, 12$ but omit ordered pairs or verification
Cold-start-style	Factors $t^2 - 17t + 60$, names both pairs, marks the answer

Same mathematics, four stage witnesses

Find all positive integer pairs (a, b) *with* $a + b = 17$, $ab = 60$. *Return the pairs and a short verification.*

Stage witness	Schematic behavior
Base response	May name $5, 12$ but omit ordered pairs or verification
Cold-start-style	Factors $t^2 - 17t + 60$, names both pairs, marks the answer
Reasoning-RL scored	Rewarded only when the full checker is satisfied

Same mathematics, four stage witnesses

Find all positive integer pairs (a, b) *with* $a + b = 17$, $ab = 60$. *Return the pairs and a short verification.*

Stage witness	Schematic behavior
Base response	May name $5, 12$ but omit ordered pairs or verification
Cold-start-style	Factors $t^2 - 17t + 60$, names both pairs, marks the answer
Reasoning-RL scored	Rewarded only when the full checker is satisfied
Final R1-style answer	Pairs and verification; schema controls reasoning visibility

Same mathematics, four stage witnesses

Find all positive integer pairs (a, b) *with* $a + b = 17$, $ab = 60$. *Return the pairs and a short verification.*

Stage witness	Schematic behavior
Base response	May name $5, 12$ but omit ordered pairs or verification
Cold-start-style	Factors $t^2 - 17t + 60$, names both pairs, marks the answer
Reasoning-RL scored	Rewarded only when the full checker is satisfied
Final R1-style answer	Pairs and verification; schema controls reasoning visibility

"The answer is 5 and 12" is mathematically right and still fails the task — being correct and being usable are different properties.

The pipeline as a branch of policies

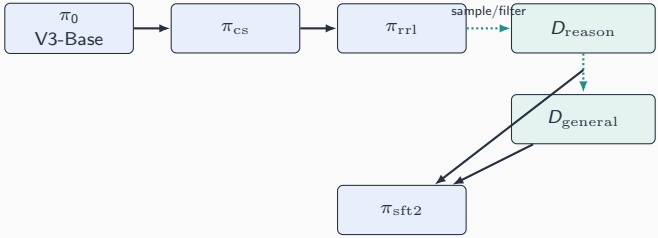
Not a chain: two roles for one base checkpoint



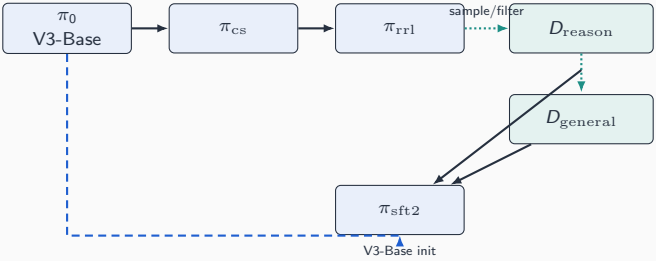
Not a chain: two roles for one base checkpoint



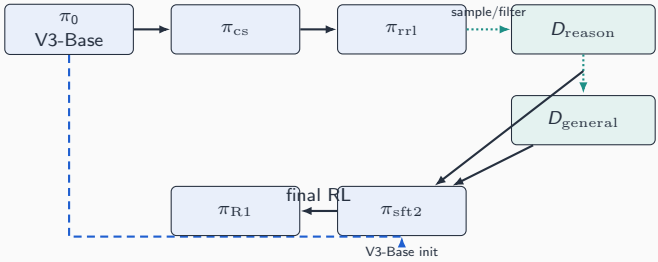
Not a chain: two roles for one base checkpoint



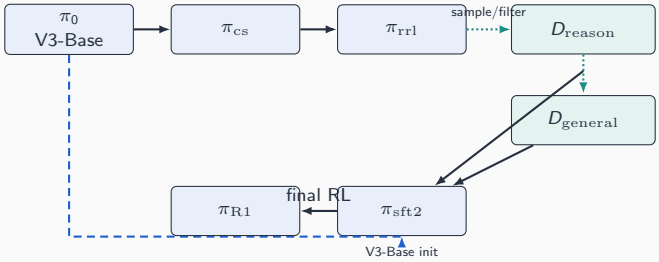
Not a chain: two roles for one base checkpoint



Not a chain: two roles for one base checkpoint



Not a chain: two roles for one base checkpoint



The dashed edge is the easily missed one: mixed SFT restarts from π_0, not from the sampler π_{rrl} that generated its training data.

Cold start, reasoning RL, and rejection sampling

Readability, then verifiable reward, then filtered data

- **Cold-start SFT**: $\pi_{\mathrm{cs}} = \mathrm{SFT}(\pi_0, D_{\mathrm{cs}})$ teaches a stable, scorable answer shape *before* RL.

Readability, then verifiable reward, then filtered data

- **Cold-start SFT**: $\pi_{\mathrm{cs}} = \mathrm{SFT}(\pi_0, D_{\mathrm{cs}})$ teaches a stable, scorable answer shape *before* RL.
- **Reasoning RL**: $r_{\mathrm{reason}} = r_{\mathrm{acc}} + r_{\mathrm{lang}}$ directly sums task accuracy with a target-language word fraction.

Readability, then verifiable reward, then filtered data

- **Cold-start SFT**: $\pi_{\mathrm{cs}} = \mathrm{SFT}(\pi_0, D_{\mathrm{cs}})$ teaches a stable, scorable answer shape *before* RL.
- **Reasoning RL**: $r_{\mathrm{reason}} = r_{\mathrm{acc}} + r_{\mathrm{lang}}$ directly sums task accuracy with a target-language word fraction.
- **Rejection sampling**: draws K candidates from π_{rrl}, keeps those with $A(o, q) = 1$ as new SFT rows — an accepted fraction like $3/8$ measures one sampler–filter pair, not general capability.

Readability, then verifiable reward, then filtered data

- **Cold-start SFT**: $\pi_{\mathrm{cs}} = \mathrm{SFT}(\pi_0, D_{\mathrm{cs}})$ teaches a stable, scorable answer shape *before* RL.
- **Reasoning RL**: $r_{\mathrm{reason}} = r_{\mathrm{acc}} + r_{\mathrm{lang}}$ directly sums task accuracy with a target-language word fraction.
- **Rejection sampling**: draws K candidates from π_{rrl}, keeps those with $A(o, q) = 1$ as new SFT rows — an accepted fraction like $3/8$ measures one sampler–filter pair, not general capability.

Reward terms and acceptance gates are different objects. Reporting a selection gate as if it were part of the reward silently conflates policy feedback with dataset construction.

Executing an R1-style loop with veRL

One logical actor, two distributed layouts

HybridFlow's controller expresses rollout, reward, and update in algorithm order; worker groups execute the distributed computation underneath.

- Rollout favors a generation layout with KV-cache room; training favors sharded parameters and optimizer state.
- Rollout records move to training workers; resharded weights move back — opposite directions, same logical actor.
- The established prompt-group key must survive that handoff: scheduling may change *where* a response runs, never *which* group it belongs to.

This is an inspectable framework case study — not a claim about DeepSeek's own private infrastructure.

From concept to code: the group GRPO normalizes over

from verl/verl/trainer/ppo/core_algos.py

```
scores = token_level_rewards.sum(dim=-1)
id2score = defaultdict(list)
for i in range(bsz):
    id2score[index[i]].append(scores[i])
for idx in id2score:
    ...
    id2mean[idx] = torch.mean(scores_tensor)
    id2std[idx] = torch.std(scores_tensor)
for i in range(bsz):
    scores[i] = (scores[i]-id2mean[index[i]])/(id2std[index[i]]+epsilon)
```

From concept to code: the group GRPO normalizes over

from verl/verl/trainer/ppo/core_algos.py

```
scores = token_level_rewards.sum(dim=-1)
id2score = defaultdict(list)
for i in range(bsz):
    id2score[index[i]].append(scores[i])
for idx in id2score:
    ...
    id2mean[idx] = torch.mean(scores_tensor)
    id2std[idx] = torch.std(scores_tensor)
for i in range(bsz):
    scores[i] = (scores[i]-id2mean[index[i]])/(id2std[index[i]]+epsilon)
```

index[i] is the prompt-group key from the previous slide, written as code: outcome rewards are grouped by it, then centered and scaled *within* the group. [INSPECTED: verl verl/trainer/ppo/core_algos.py]

Distillation and release evaluation

Six artifacts, one optimizer split, one deployment ladder

- R1-Distill learners get *SFT only* on the curated 800k set — no learner-side RL. Their name marks data lineage, not RL repetition.
- Reported 32B comparison: SFT on R1-generated data *outscores* direct large-scale RL on the same base (AIME pass@1 72.6 vs. 47.0). [REPORTED: DeepSeek-R1 paper]
- Local deployment ladder: three artifacts all reach CUDA generation; none of their capped outputs contain a parseable final answer — admission is not capability.

No local row is an R1-teacher comparison: the official 671B teacher exceeds the verified four-H200 capacity this book's local work used.

Synthesis: R1 versus R1-Zero

Two purposes, not two competitors

Axis	R1-Zero	R1
Purpose	Scientific demonstration	Practical release and recipe
Starting point	Base model, pure RL	Cold-start-shaped base
Release target	Emergent behavior, studied	Readable, schema-controlled answer

Main risk in reading either side: treating R1-Zero's traces as magic, or treating R1's public recipe as if it exposed every private detail.

Concept-to-code map for the chapter

- **public stage sequence and reward design** → `DeepSeek-R1/DeepSeek_R1.pdf`
- **released artifact family and licenses** → `DeepSeek-R1/README.md`
- **prompt-group grouping inside GRPO** → `verl/verl/trainer/ppo/core_algos.py`
- **executable R1-style recipe (under development)** → `verl-recipe/r1/`

Readable, verifiable, and released are three different obligations.

Next: the math and prover chapter continues the verification ladder this chapter's checker-based rewards depend on.