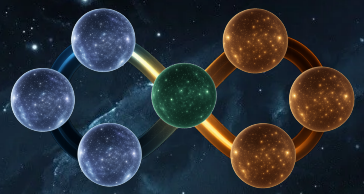


Multiverse: Your Language Models Secretly Decide How to Parallelize and Merge Generation

Xinyu Yang[1]*, Yuwei An[1]*, Hongyi Liu[1],
Tianqi Chen[1,2], Beidi Chen[1]

[1]Carnegie Mellon University , [2]NVIDIA

Motivation: I/O-bounded AR Generation

Refer to Kinetics, if we want to generate one token, we need:

1. linear modules computation: $2NP$
2. self-attention computation:$2rNL_{in}D + rND$
3. KV access: $2INL_{in}D + IND$ (do not share prefix in the middle of generation)
4. Parameter access: $2IP$

N: batch size; P: parameter; I: Arithmetic Intensity; r: Group size; D: hidden dimension

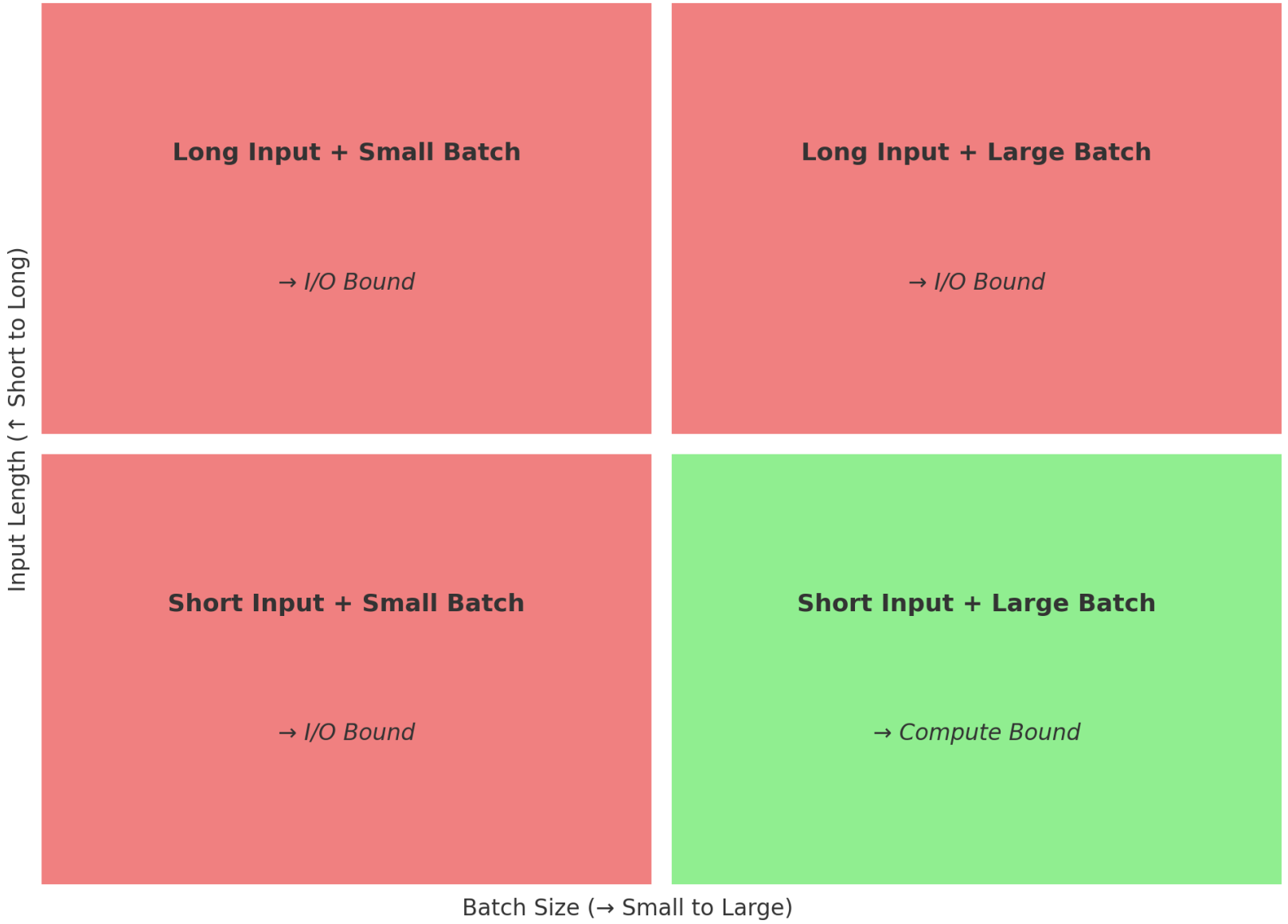
Small Batch Scenarios : If $N = 1, L_{in} = 4096, \mathrm{I} = 562.5,\ P = 32\mathrm{B}, \mathrm{r} = 4,\ D = 4096$, Parameter access occupies 99% time.

If we want to be compute-bound, we need to have a batch size of 756

If $\mathrm{L_{in}} = 32768$, KV access dominates, it is never compute-bound

Motivation: I/O-bounded AR Generation

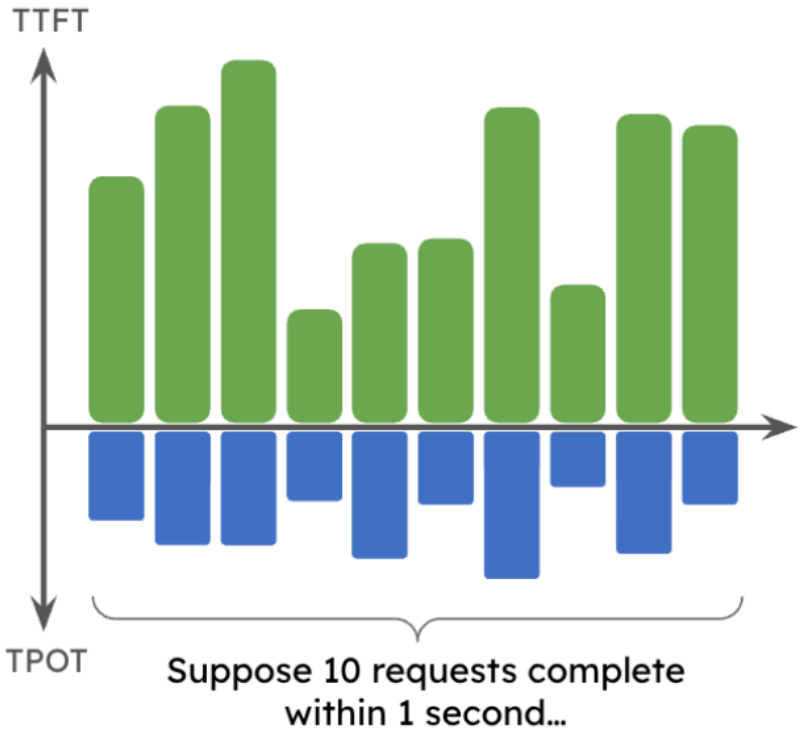
Therefore, in most reasoning tasks. We are facing I/O-bound. This is very different to our current server systems optimizing for large throughput chatbot settings, where most of the cases fall into Short Input + Large Batch.



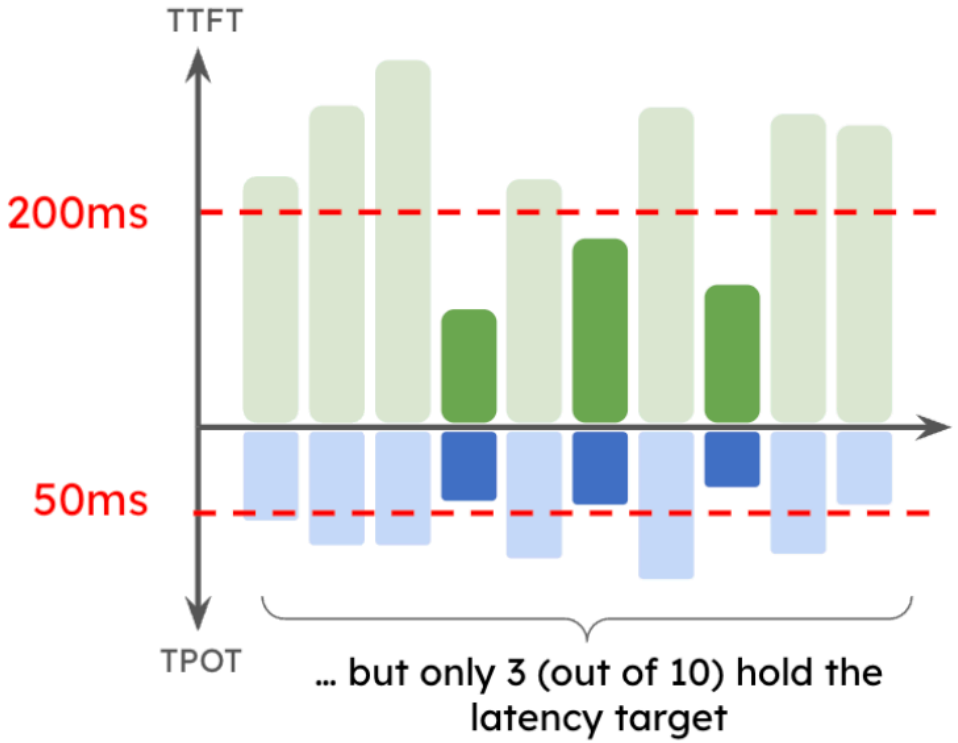
Motivation: Throughput is not everything

Many users care about their own latency instead of the overall throughput. This is especially important when you want to solve complex tasks.

Throughput = completed request / time
= **10** req / s



Goodput = completed requests **within SLO** / time
= **3** req / s



Goal 1: We want parallel generation.

From the above analyses, we find that the generation speed/latency is actually bottlenecked by the autoregressive generation. Therefore, can we enable fine-grained parallel generation within one example?

1. Since most generation cases are I/O-bounded, this do not increase inference time with parameter and KV sharing.
2. When we care about latency or goodput, parallel generation will reduce the sequential generation steps, resulting in efficiency gain.

As many work exploring this direction, why they do not solve this problem?

1. For diffusion models, they brute-force predict the output for the whole sequence at each step, requiring many denoising steps. Moreover, none of them except Gemini Diffusion works for real-world reasoning tasks.
2. For methods like Best-of-N/MCTS/ToT, they relies on external tools.

Goal 2: we want adaptive, internal parallel generation

Based on the limitations of past work, we find that diffusion models focus on the token-level parallelism, while other methods will focus more on semantic-level parallelism.

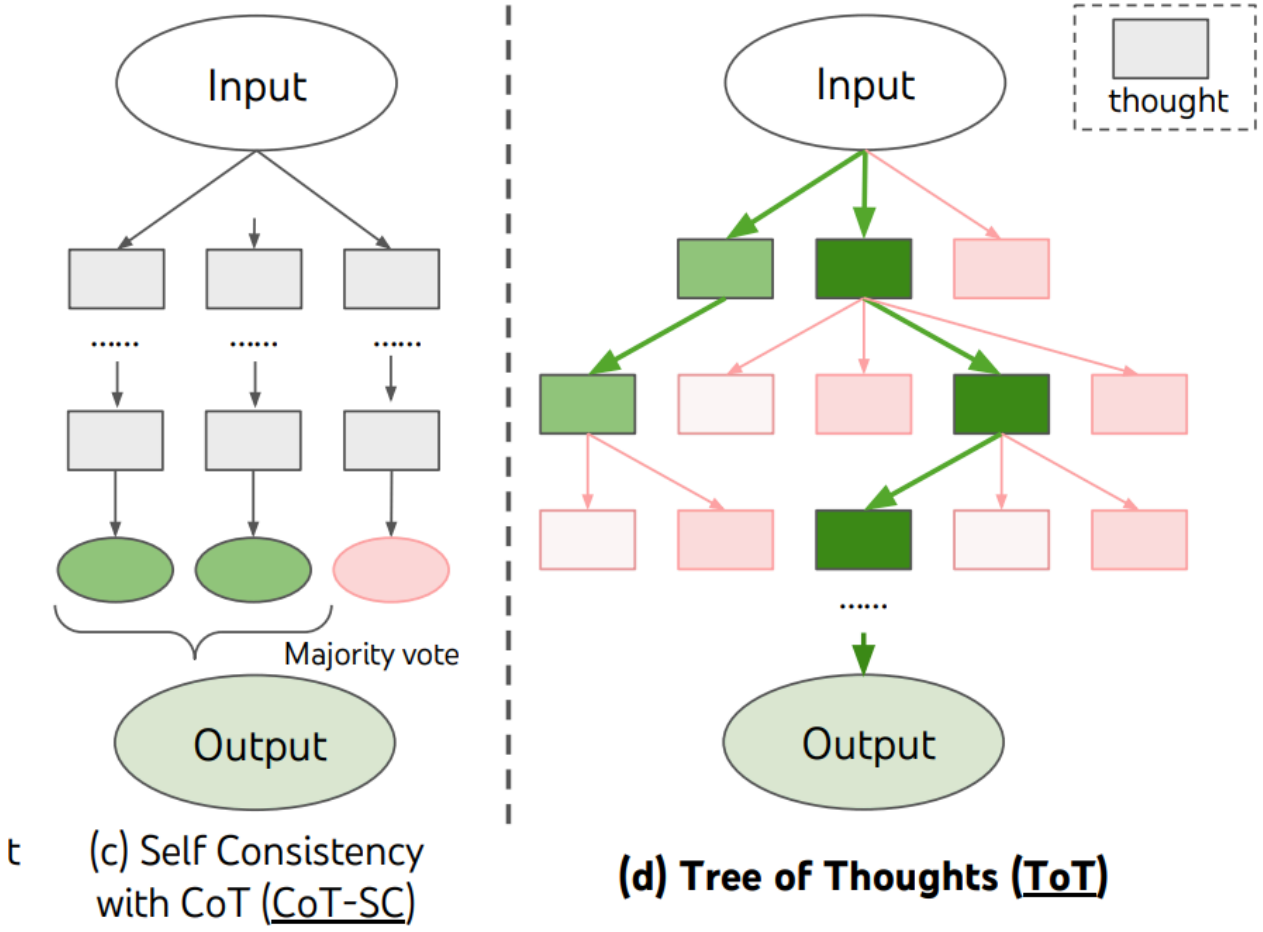
token-level

semantic-level

Current modeling works are mainly from token-level. In this work, we focus on semantic-level parallelism, exploring how to adaptively enable it in the model architecture.

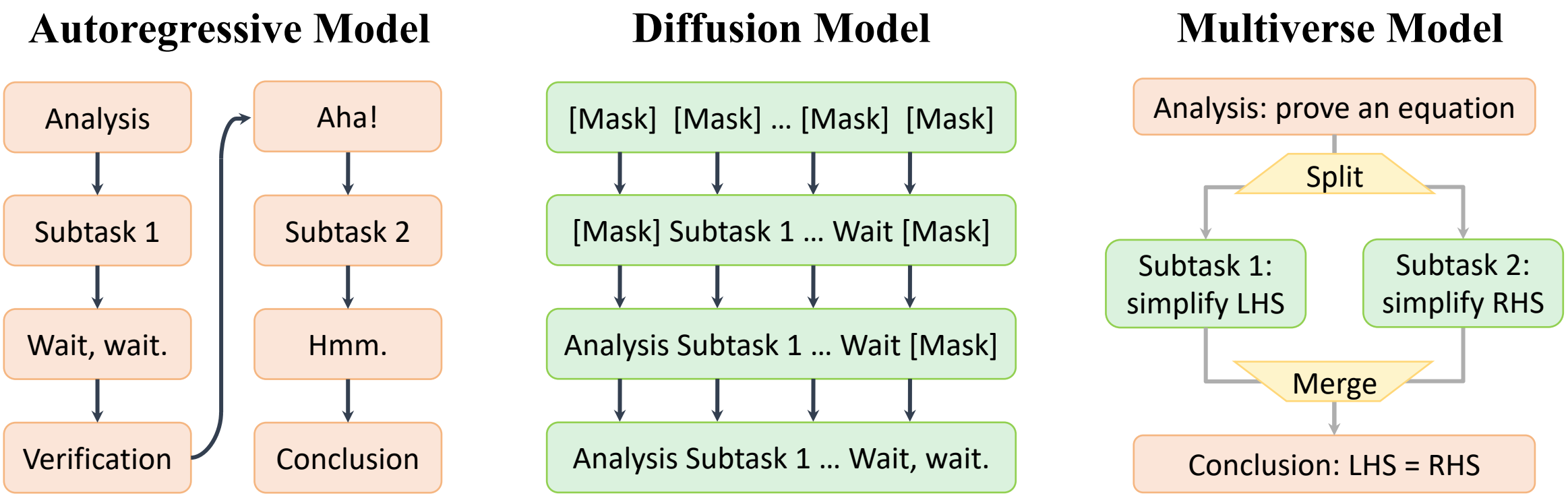
What is needed to enable this ability?

In past methods, we find most generation steps are still AR steps. Therefore, we only need to enable the model to parallelize its generation at very few steps. This is supported by prefix cache. However, we need to consider how to natively merge them without information loss.



Solution: Multiverse based on MapReduce paradigm

It natively enable parallel generation by introducing map stages for task splitting and reduce stages for result merging

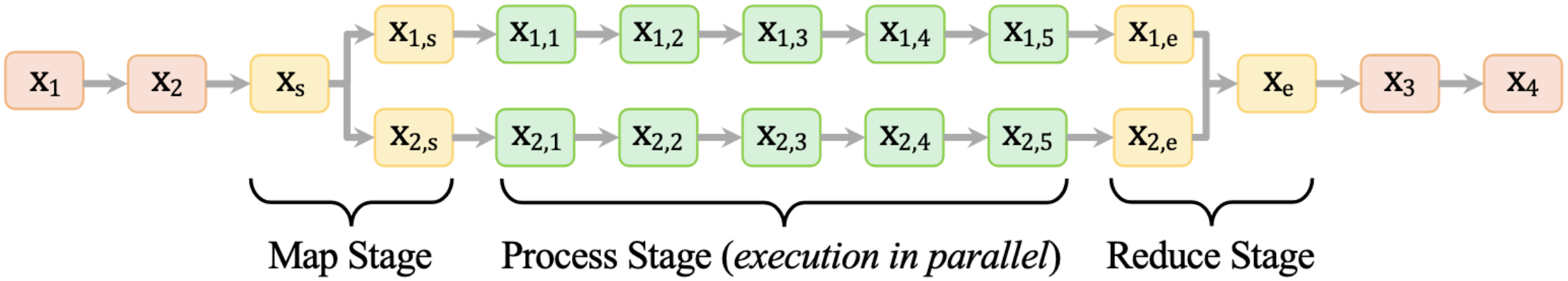


Note: Multiverse is just a proof-of-concept for now, do not expect too much speedup on the current model

Question 1: How to ensure difference between paths

If we just initial two paths with the same prefix, it is just repeated sampling.

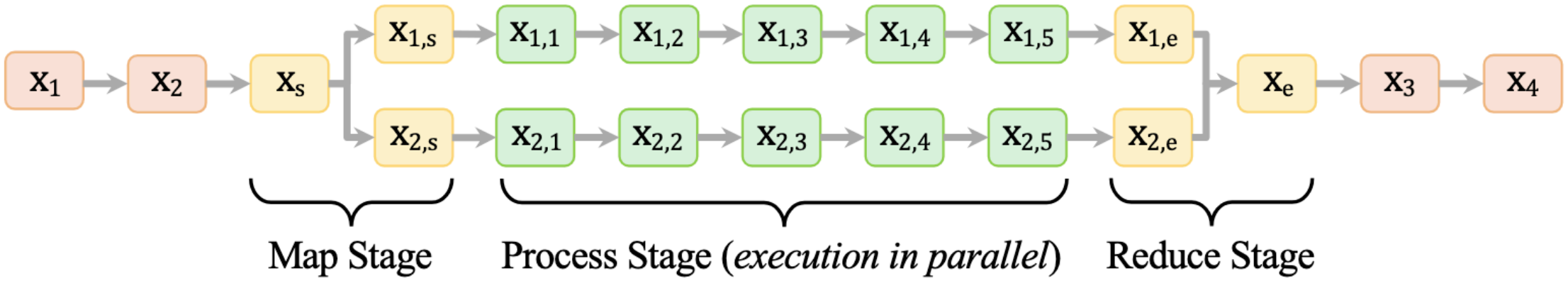
As a result, they cannot execute two different tasks.



Our solution is very simple: since the AR generation is dependent on its prefix. If their prefix is not completely same, they should be different in generation. This is commonly used in sharing same system prompt for different questions. Therefore, we only need to first define each task sequentially and then prefill each path with its own prefix indices as hint.

Question 2: How to merge information from paths

While the map stage do not break the AR modeling (you just have two independent paths), the reduce stage requires us to merge these paths



We also use a simple way to modelling this: Just let the future generation conditioned on every tokens before it. This means, at the same time, we can have more than one things happen.

Reduce Stage. After completing all branches, Multiverse shift back to sequential generation to conclude them, which is conditioned on preceding tokens from all branches, modeled as $P(\mathbf{x}_{e,[3:4]}|\mathbf{x}_{1,[s,1:6,e]}, \mathbf{x}_{2,[s,1:6,e]}, \mathbf{x}_{[1:3,s]})$.

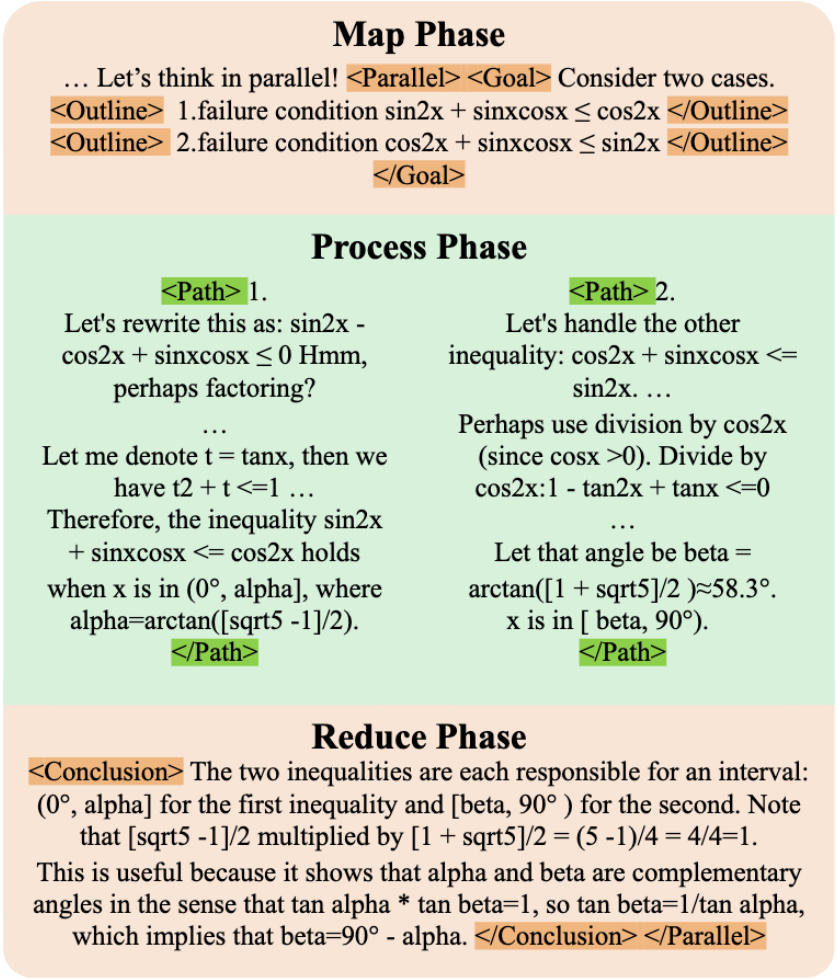
Question 3: How to switch between stages?

The idea is good. But when do I know that I should move to a different stage

Inspired by the recent reasoning models that use <Think> and <Answer> to separate their stages. We think that we can use special tokens to indicate the start or end of a stage.

For example, if we see </Goal>, we will know that we should prefill each <Path> X and split the current request into multiple requests based on the count of <Outline> with different task id.

Therefore, we need to have an XML-structure to automatically control its generation workflow. Moreover, nested structures are also supported.



Observation: the existence of intrinsic parallelism

The modeling framework is good, but do we have such parallel data?

… The ratio of the areas of two regions is given as 2:3 or 3:2.
Case 1: $A_1/A_2 = 2/3$
… Substitute $b_1 = b_2 + 100$:
… $b_1 = 75 + 100 = 175$
Case 2: $A_1/A_2 = 3/2$
… Substitute $b_1 = b_2 + 100$:
… $b_2 = -175$, which is not true since length is positive.
So, we have $b_1 = 175$ and $b_2 = 75$

… Let me go through each system one by one and check if they're possible based on my astronomy knowledge.
System 1: …
System 2: …
System 3: …
System 4: …
System 5: …
So now, let's go back to the systems: …

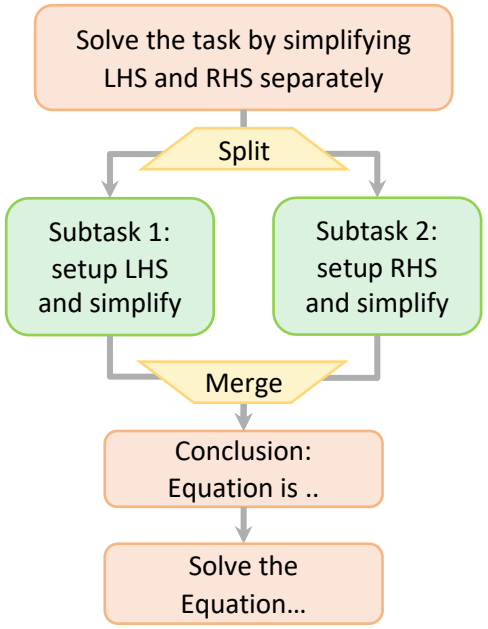
Assume n=1. Moves are taking 1 stone or a prime number of stones or a positive multiple of 1 stone (any number of stones). …
Consider n=2. Moves are taking 1 stone, or a prime number of stones, or a positive multiple of 2 stones. …
Consider the case when n is large …

Wait, this system is: …
This might be too hard.
Alternatively, assume a=b …
Alternatively, coefficients …
Alternatively, use t=1 …
Alternatively, use the value of t found earlier when assuming t=-1+sqrt(11)≈2.316.
Wait, different problem …
Alternatively, assuming that a=1/2: …

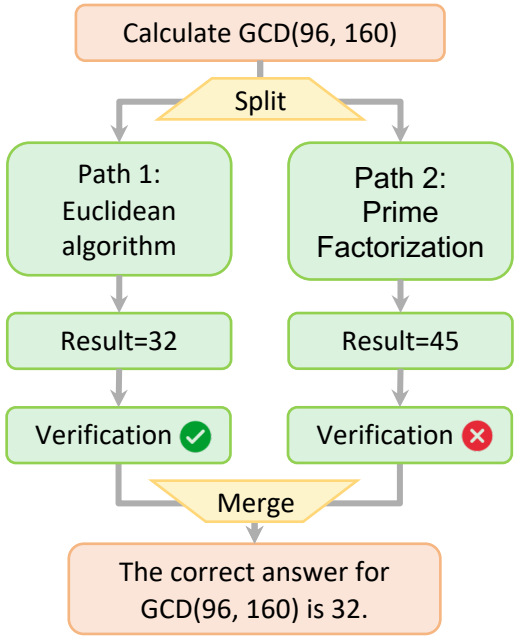
We find their common existence in CoT traces for reasoning tasks

Observation: the existence of intrinsic parallelism

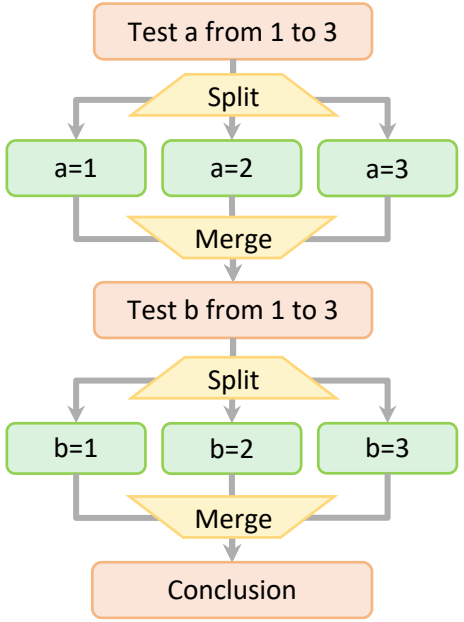
They can be typically classified into two categories, which can appear alone or be combined into a consecutive or recursive structure.



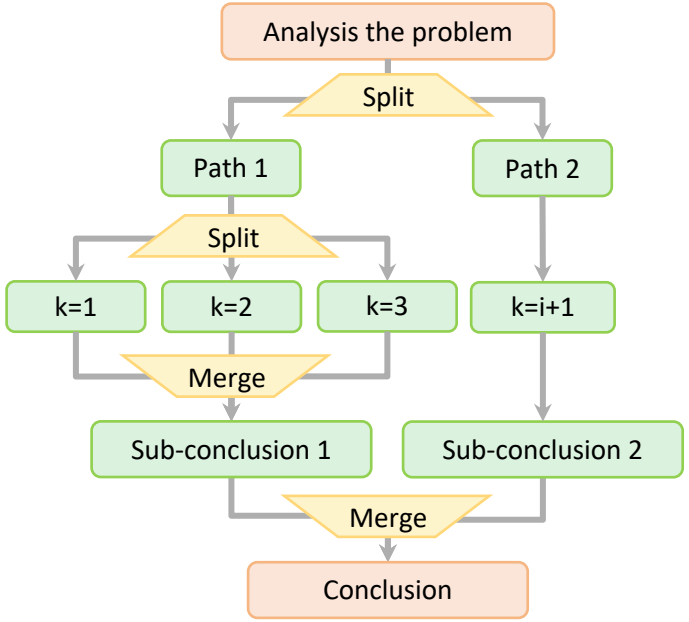
1. Collective Branch



2. Selective Branch



3. Consecutive Structure



4. Recursive Structure

Observation: the existence of intrinsic parallelism

Next, we prompt Gemini-2.5 Pro to analyze the occurrence of different types of parallelism within the CoT traces generated by the s1 team.

	Collective Branch			**Selective Branch**		**Total**
	Case Study	Subtask Execution	Other	Path Exploration	Other	
Deepseek R1	28.0 | 2.00	47.3 | 3.38	3.2 | 0.23	19.4 | 1.39	1.1 | 0.07	99.0 | 7.07
Gemini 2.0 Flash	39.4 | 2.82	45.0 | 3.22	2.9 | 0.21	8.3 | 0.60	1.4 | 0.09	97.0 | 6.94

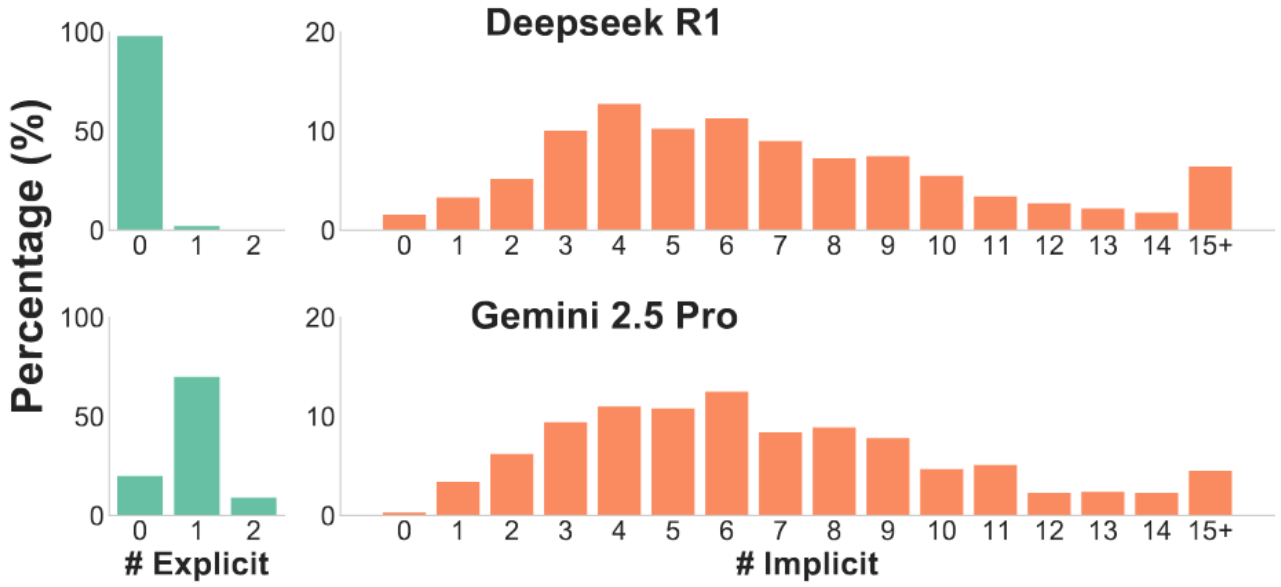
First, 98% of data exhibit parallelism in generation, with an average frequency of 7 times in each example, showing the common existence of parallelism.

Additionally, since we only analyze it in each single trace. More cases fall into the collective branches where all branches work on some important subtasks

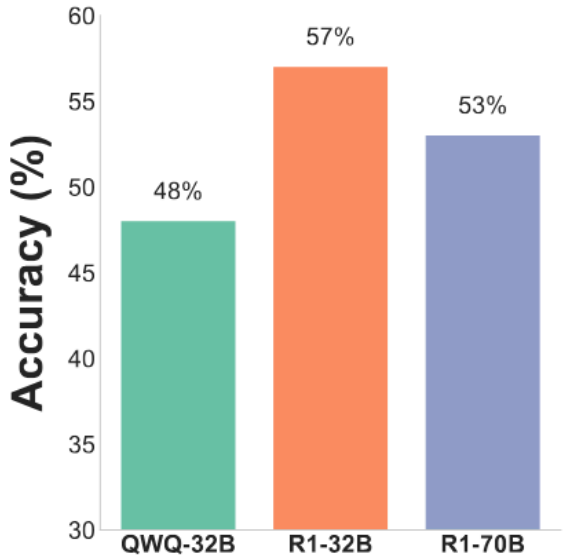
Observation: intrinsic to explicit parallelism

However, the existence of intrinsic parallelism do not directly reflect to our desired MapReduce structures, how can we add this structure to the trace?

We first prompt the model with hints to think following our structure. However, since the model still generates sequentially, it seldomly output such structure. **This indicates it is hard to directly generate such CoT traces use AR models**



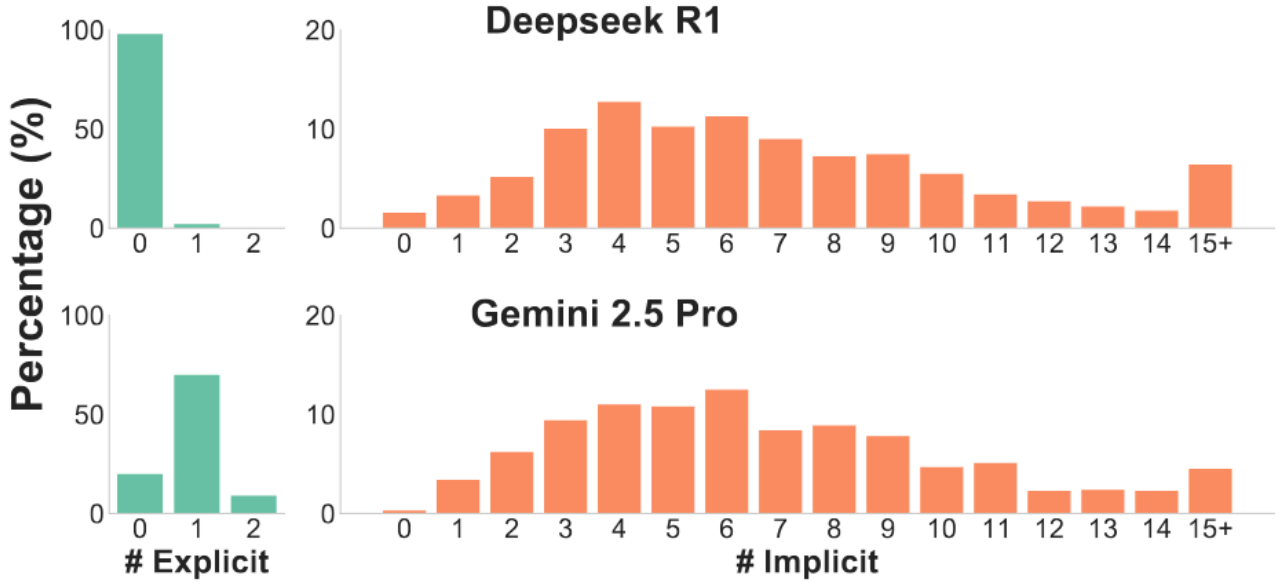
(a) Prompting Test: Comparing Explicit and Implicit Structure Counts



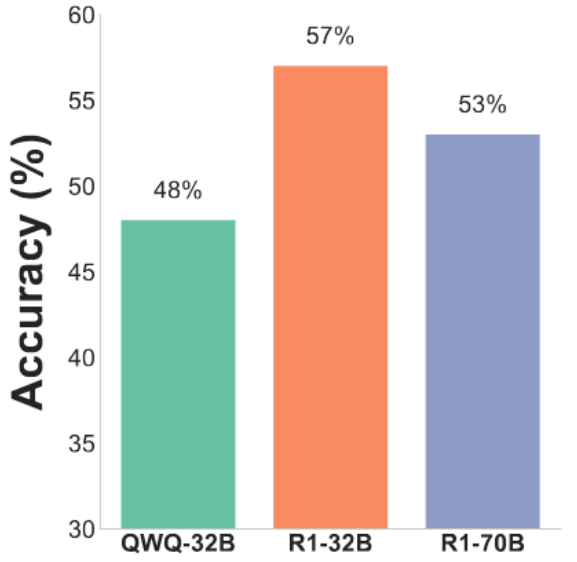
(b) Probing Test: Classifier Accuracy

Observation: intrinsic to explicit parallelism

We further use a probing test to check whether the model itself is aware of the starting of parallelizable branches. Unfortunately, existing models not have such a sense. we hypothesize the parallelizable branches are just generated following some internal parallel pattern in the data. **The model do not understand it.**

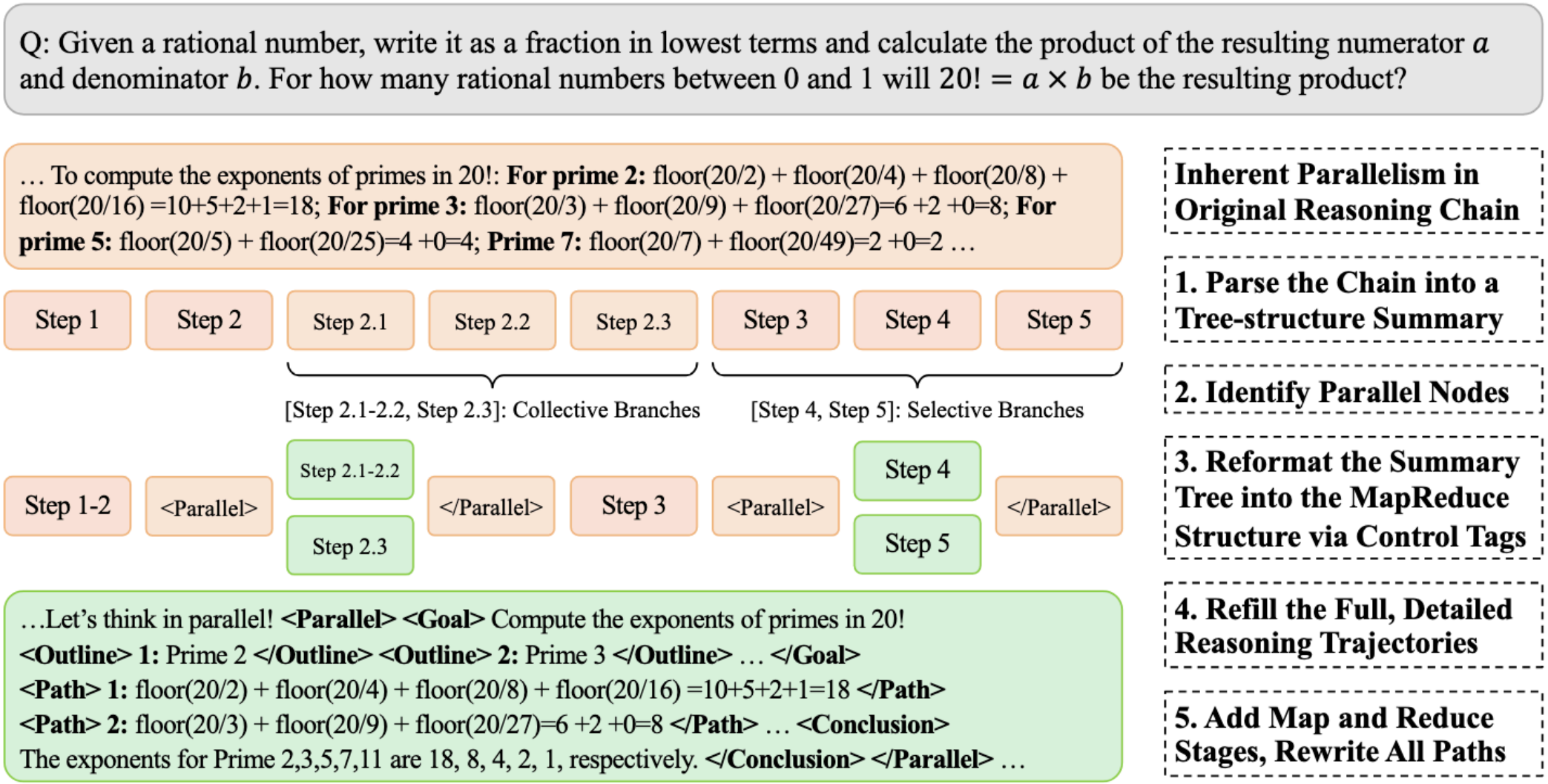


(a) Prompting Test: Comparing Explicit and Implicit Structure Counts



(b) Probing Test: Classifier Accuracy

Solution: Rewrite the unstructured CoT traces



(a) Multiverse Curator automatically generated **Multiverse-1K** using an LLM-assisted data curation pipeline.

Solution: Rewrite the unstructured CoT traces

Please refer to our Appendix for the detailed prompt, we want to mention some important observations in this prompting framework:

1. Do not fuse several tasks into our prompt, this will reduce accuracy.
2. LLMs are not good at copy-and-paste as they will tend to skip some sentences during long context generation. Therefore, we find explicitly prompt with “sentence by sentence” is very helpful (but requires more thinking time). Moreover, we will restrict the edit distance to smaller than 0.2 to avoid skipping too much sentences.
3. Reasoning LLMs will tend to have words “Similarly”, “Alternatively”, or use cross reference in each paths. Therefore, we will ask the assistant LLMs to rewrite each path to avoid such problems

Note: most of the data quality is preserved, but we still need mixing sequential data for better performance

Problem: how to instantiate our modeling

For algorithm design, we need to incorporate Multiverse modeling into existing token mixing layers, which means that we need a variant of attention layer ensure that different paths can be executed in parallel.

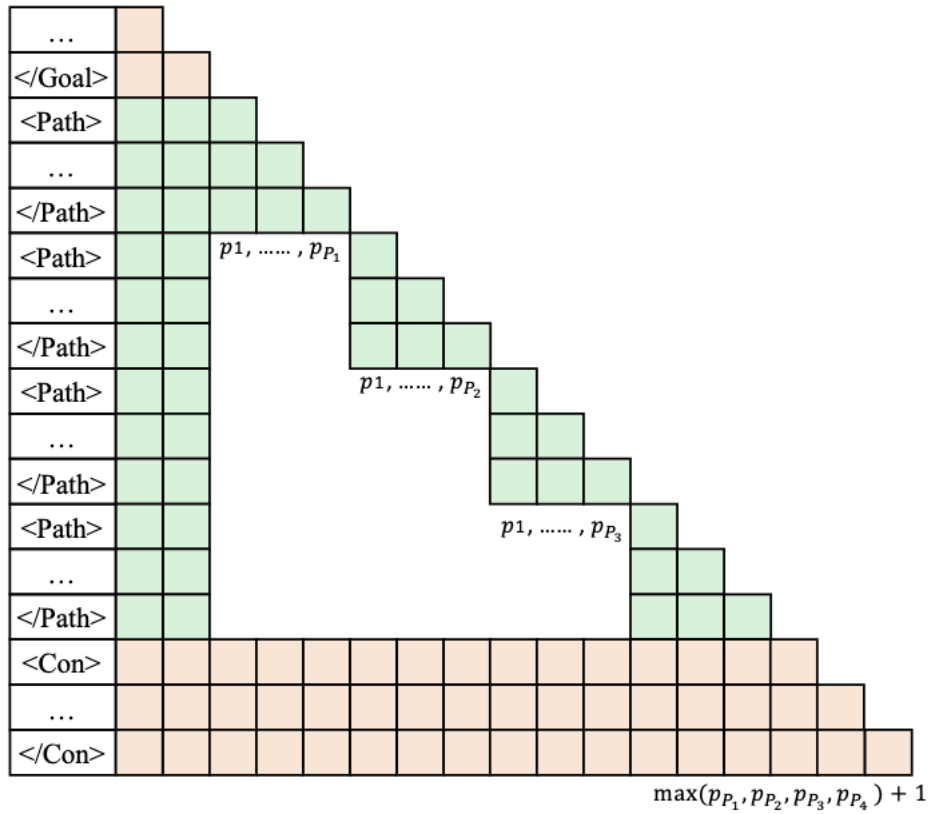
This is not hard to achieve, requires:

1. modifying attention masks: different paths should not see each other

2. modifying position ids: different paths should start from the same position

For the end of paths, we will use the max positions from all paths for future generation.

So Multiverse actually let the model to decide its own sparse attention pattern

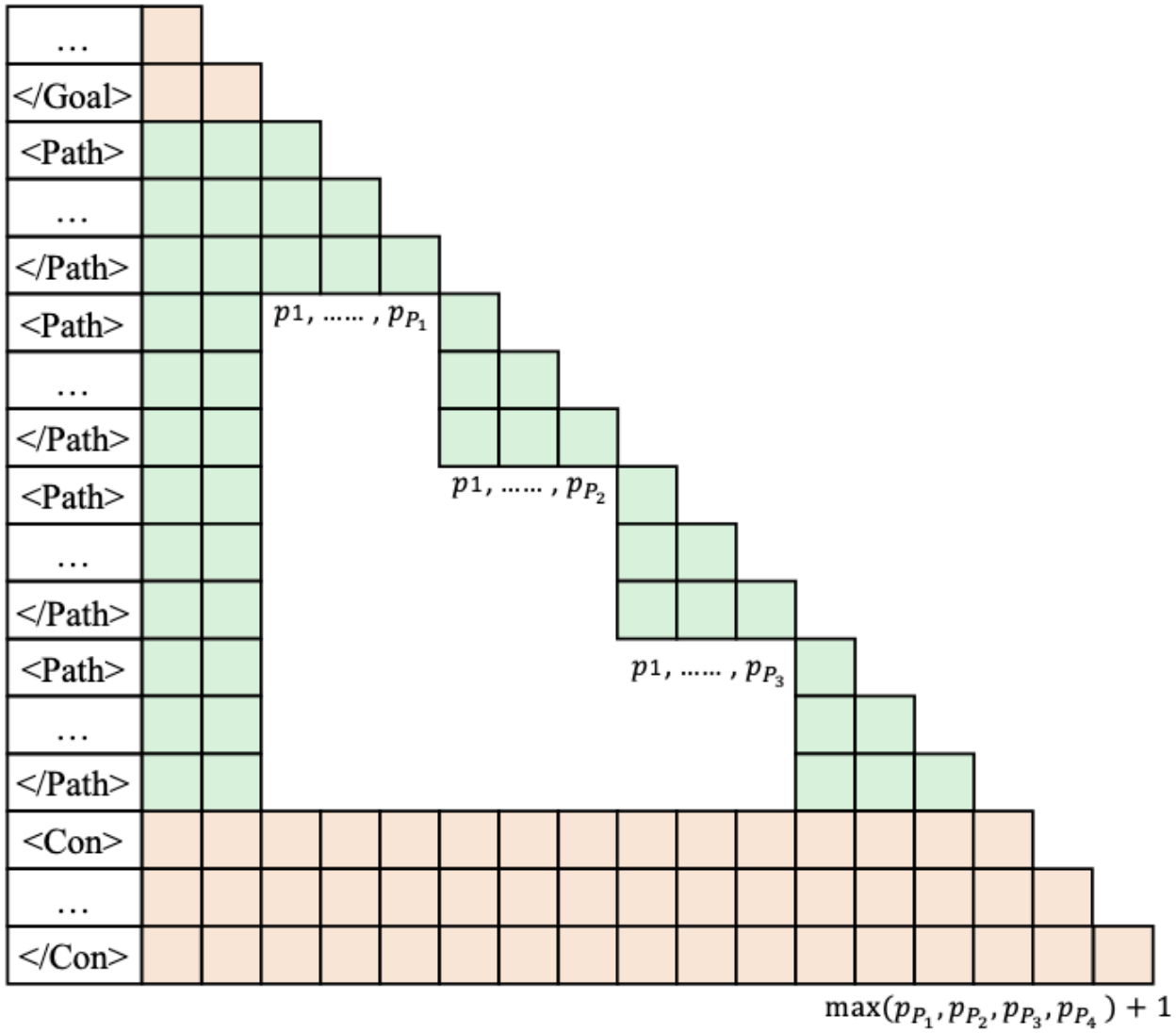


Practical Advantages of Multiverse Attention

First, it can be represented with the following attention mask and positions. Therefore, it can be trained in parallel using customized attention mask.

Moreover, since it is very similar to the original casual mask, you can convert an AR model into a Multiverse model with training only on a few examples, which shows its data efficiency.

Create a Multiverse model can be cheap without pre-training

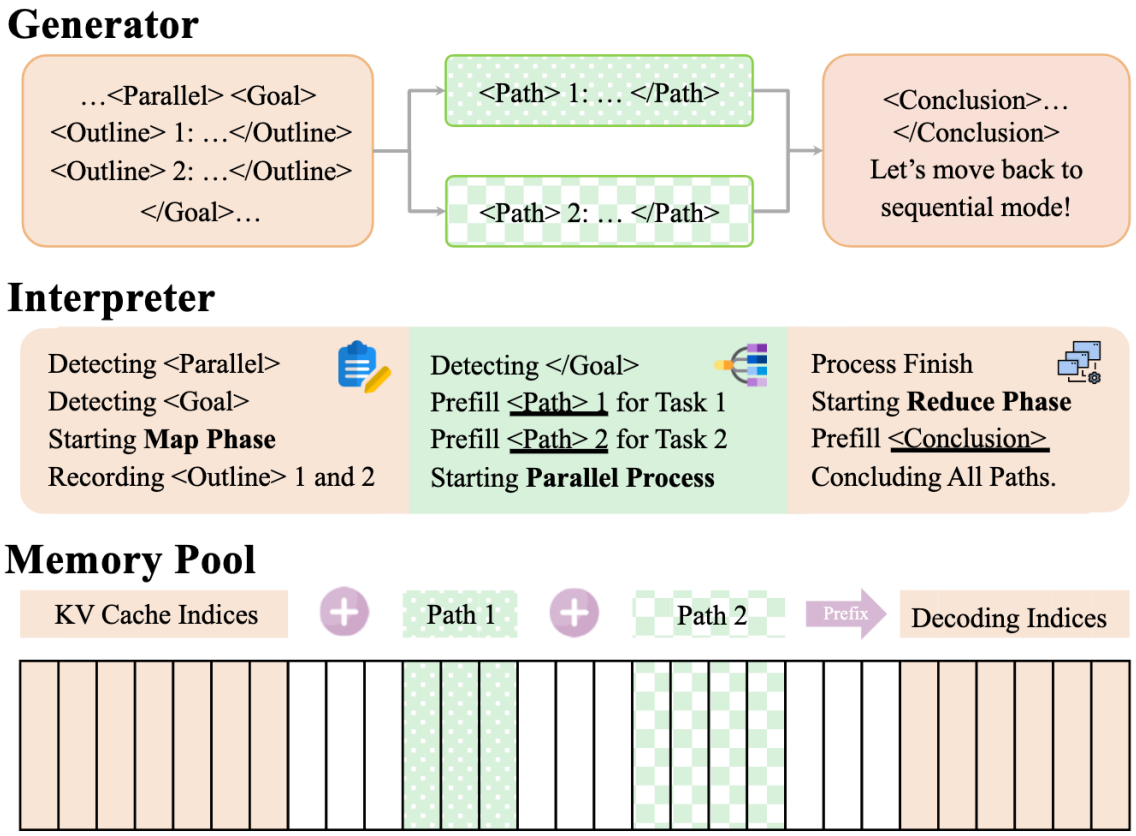


Problem: how to generate in parallel

The final technical challenge is existing inference engines cannot support Multiverse model due to its complex generation flow. To address , we introduce Multiverse engine based on SGLang.

We choose SGLang instead of vLLM due to three reasons:

1. radix attention for prefix sharing in parallel generation
2. continuous batching is very important for us to switch between different stages
3. SGLang supports the page size of KV memory to be one



Train a Multiverse models

Based on Multiverse Curator, we generated Multiverse-1K by rewriting the reasoning traces generated by Deepseek R1. Typically, this can be done in 1 day (based on how much quota you have)

We find that when distilling small models (32B), Deepseek R1's data is much better than that from Gemini and Claude, with the later models result in significant repeating and performance drop. We hypothesize this maybe caused by low entropy or the huge gap of model sizes as you cannot expect a small model to think the same as a very large model.

Next, we perform supervised fine-tuning with our Multiverse attention. Before training on each example, we will generate our masks and positions by scanning the context from left to right as a DAG structure (similar to our inference). For further acceleration, we can consider to preprocess them before training. Typically, our training only takes 3 hours on 8 B200 GPUs using flex attention.

Train a Multiverse models: more details

To preserve the data quality, we train our model for 8 epochs, with a mixture ratio (AR: Multiverse) of 1, 1, 0.9, 0.5, 0.5, 0.1, 0, 0. We will form a pair for each example and use a symmetric way to sample our training data.

We further try to use separate prompts for different sources:

1. AR: Think step by step before answering.
2. Multiverse: Think step by step and in parallel before answering.

From our experiments, this just slightly improve the degree of parallelism during inference. Therefore, more explorations need to be done along controllability.

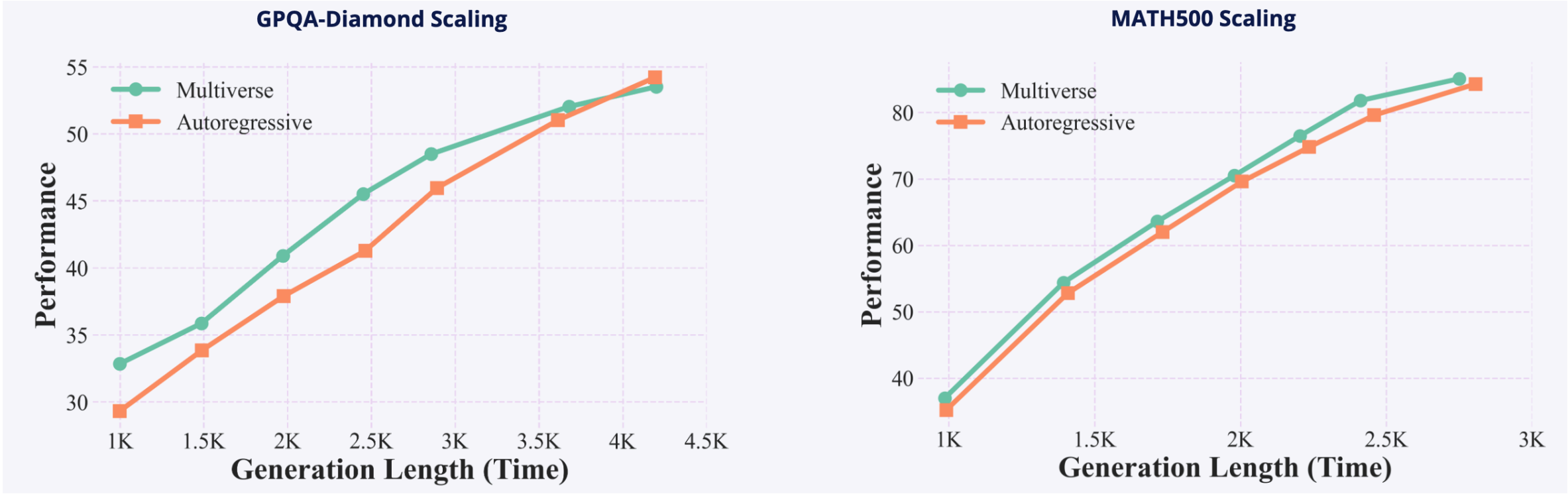
Evaluation: real-world reasoning ability

Multiverse-32B achieves significant improvements over the Qwen2.5 model by 24.5% after SFT on Multiverse-1K, while matching or exceeding the performance of AR-LLMs. This indicates the effectiveness of our modeling, showing the modification do not hurt model performance.

	AIME24		AIME25		MATH500		GPQA-Diamond	
Model / Metric	pass@1	# parallel	pass@1	# parallel	pass@1	# parallel	pass@1	# parallel
s1-32B	35.4	1.00	25.8	1.00	88.6	1.00	48.0	1.00
s1.1-32B	52.9	1.00	41.7	1.00	93.4	1.00	62.6	1.00
Qwen2.5-32B-Instruct	15.8	1.00	10.4	1.00	80.4	1.00	47.0	1.00
Autoregressive-32B	**54.6**	1.00	<u>45.0</u>	1.00	**92.8**	1.00	<u>61.6</u>	1.00
Multiverse-32B-zero	52.1	1.04	44.2	1.05	<u>92.4</u>	1.12	**63.6**	1.17
Multiverse-32B	<u>53.8</u>	1.18	**45.8**	1.15	91.8	1.15	60.7	1.17

Evaluation: efficient test-time scaling

Multiverse-32B exhibits a superior tradeoff between performance and latency than AR-LLMs. It achieves this by generating more tokens within the same wall-clock time. (We evaluate on GPQA and MATH500 since they are more sensitive to the context length)



Thanks You!

Q&A