Empirical Investigation of Neural Symbolic Reasoning Strategies

Yoichi Aoki,[1] **Keito Kudo,**[1] **Tatsuki Kuribayashi,**[1,2] **Ana Brassard,**[3,1]
Masashi Yoshikawa,[1] **Keisuke Sakaguchi,**[1,3] **Kentaro Inui**[1,3]

[1]Tohoku University, [2]Langsmith, Inc., [3]RIKEN

{youichi.aoki.p2,keito.kudo.q4}@dc.tohoku.ac.jp,
kuribayashi@tohoku.ac.jp, ana.brassard@riken.jp,
{yoshikawa,keisuke.sakaguchi,kentaro.inui}@tohoku.ac.jp

Abstract

Neural reasoning accuracy improves when generating intermediate reasoning steps. However, the source of this improvement is yet unclear. Here, we investigate and factorize the benefit of generating intermediate steps for symbolic reasoning. Specifically, we decompose the reasoning strategy w.r.t. step granularity and chaining strategy. With a purely symbolic numerical reasoning dataset (e.g., A=1, B=3, C=A+3, C?), we found that the choice of reasoning strategies significantly affects the performance, with the gap becoming even larger as the extrapolation length becomes longer. Surprisingly, we also found that certain configurations lead to nearly perfect performance, even in the case of length extrapolation. Our results indicate the importance of exploring effective strategies for neural reasoning models. [1]

1 Introduction

Artificial intelligence researchers have been attempting neural-symbolic integration for a long time (d'Avila Garcez and Lamb, 2020; Hamilton et al., 2022). Neural models tend to perform better when generating intermediate reasoning steps in addition to the answer. This phenomenon was seen across various reasoning tasks, such as math word problems (Wei et al., 2022; Cobbe et al., 2021; Kojima et al., 2022; Recchia, 2021; Lewkowycz et al., 2022), commonsense reasoning (Wei et al., 2022; Wang et al., 2022), and symbolic reasoning (Wei et al., 2022; Kojima et al., 2022). However, it is yet unclear which factors in the intermediate step generation bring the benefit. Previous studies often used different strategies for step generation in an ad-hoc manner. To investigate this, we break down the neural reasoning process into two strategies: *output strategy* and *chaining strategy* (Figure 1). The output strategy (§2.1) determines the granularity of
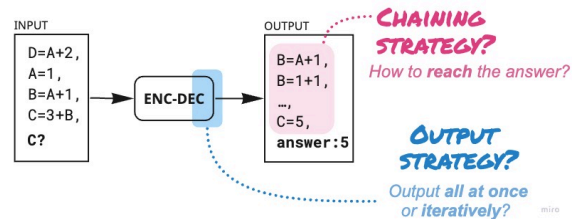

Figure 1: In a controlled setting, we found that output and chaining strategy choice significantly impact performance when conducting multi-step reasoning.

intermediate reasoning step generation (all at once vs. step-by-step vs. token-by-token). Some studies trained the models to generate reasoning steps and a conclusion derived from them at once (Nye et al., 2021; Lewkowycz et al., 2022; Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022; Recchia, 2021), some generated a single reasoning step given the input and iterated this process until achieving a conclusion (Sanyal et al., 2022; Picco et al., 2021; Tafjord et al., 2021), and others iteratively generated sub-goals as well as reasoning steps (Liang et al., 2021; Shwartz et al., 2020).

In turn, the chaining strategy (§2.2) defines the reasoning path direction (shortest path vs. exhaustive path vs. backward path). For example, some studies used a backward chaining process (Picco et al., 2021; Rocktäschel and Riedel, 2017; Cingillioglu and Russo, 2019), while others adopted exhaustive searches (Tafjord et al., 2021; Liang et al., 2021; Yang et al., 2022).

To compare the strategies, we prepared a test bed of numerical reasoning problems in a simplified language (Figure 1). This format allows for more controlled testing while serving as a necessary condition—should a model fail to solve it, it cannot be expected to adequately generalize to more complex math word problems.

We found that both strategies substantially affect the symbolic reasoning performance of neural seq2seq learners. Overall, iterative generation

---

[1]Code available at: https://github.com/ao1neko/reasoning-strategy

"ALL AT ONCE"

INPUT ──[ENC-DEC]──▶ **B=A+1,B=1+1,…,C=5, answer:5**

"STEP BY STEP"

INPUT ──[ENC-DEC]──▶ **B=A+1**

INPUT, **B=A+1** ──[ENC-DEC]──▶ B=1+1

⋮

INPUT, B=A+1, ..., C=5 ──[ENC-DEC]──▶ **answer:5**

"TOKEN BY TOKEN"

INPUT ──[ENC-DEC]──▶ **B**

INPUT, **B** ──[ENC-DEC]──▶ =

⋮

INPUT, B=A+1, ...swer: ──[ENC-DEC]──▶ **5**

(a) *All-at-once*: output the entire reasoning chain and answer in a single call. *Step-by-step*: iteratively build the output with a single calculation step per call. *Token-by-token*: iteratively output only one *token* per call.
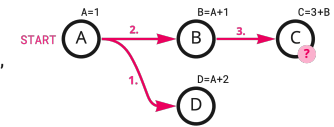
"SHORTEST PATH"
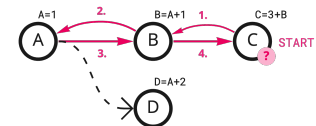
B=A+1, B=1+1, B=2, C=3+B,
C=3+2, C=5, answer: 5
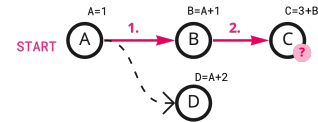
"EXHAUSTIVE"

D=A+2, D=1+2, D=3, B=A+1,
B=1+1, B=2, C=3+2, C=5,
answer: 5

"BACKWARD"

C=3+B, B=A+1, A=1, B=1+1,
B=2, C=3+2, C=5, answer: 5

(b) The graph nodes represent variables and edges their dependencies. *Shortest path*: a minimal chain starting from the first necessary equation. *Exhaustive*: greedily solve all equations until the target is reached. *Backward*: start from the target's equation, backtrack over dependencies until a known value is reached, then solve each equation in order.

Figure 2: Overview of (a) output and (b) chaining strategies given the INPUT: D=A+2, A=1, B=A+1, C=3+B, C?

outperformed all-at-once outputting, and roughly granular reasoning steps (i.e., shortest-path chaining) lagged behind finely granular steps (i.e., exhaustive and backward chaining). Surprisingly, some settings had near-perfect performance even in generalization tests which extrapolate over greater reasoning depths and unseen numbers during training.

2 Experimental settings

Problem definition. We evaluated the models' ability to iteratively perform arithmetic operations over given symbols. Given a series of equations, the task is to answer the value of a target variable (Figure 1). Each question also has a certain reasoning depth—the number of *necessary* equations to reach the answer. For example, the depth of the question A=1, B=2+A, C=3+B, D=2, C? is 3 (A=1, B=2+A, C=3+B).

Each equation defines either an assignment (e.g., A=1) or a modular addition and an assignment (e.g., B=3+1). The addition is mod 100. The question contexts also contain distractors that are not necessary to calculate the answer (e.g., D=A+2 in Figure 1). A value assigned to a particular variable is typically referred to in different equations (e.g., A=1, B=A+1). Numbers, variables, and the ordering of equations are randomly assigned.

Motivation for using artificial data There are mainly three advantages to this dataset. First, the

symbolic format allows easier control of reasoning depth for generalization tests. Specifically, we trained a model using instances with shallow (1-5) depths and evaluated them with instances with shallow/deep (1-12) depths. On the other hand, math word problems are harder to control for reasoning depth (e.g., it is not easy to come up with various instances which have a reasoning depth of 10). Second, we wanted to avoid the "spurious bias" that natural (math word) texts implicitly bring into the model (Gururangan et al., 2018; Gupta et al., 2021; Al-Negheimish et al., 2021; Sugawara et al., 2018; Jia and Liang, 2017; McCoy et al., 2019). Third, we assume that our setting is the necessary condition for solving math word problems. It is unreasonable to expect that a model that can't solve this pure numerical reasoning task can solve more complex tasks.

In total, we prepared 5K instances for training and 2.4K for testing.

2.1 Output strategies

We compared three configurations: all-at-once, step-by-step, and token-by-token (Figure 2a).

All-at-once: The model outputs the entire reasoning chain and the final answer in a single call (i.e., *chain-of-thought* style) (Wei et al., 2022; Cobbe et al., 2021; Yavuz et al., 2022; Shwartz et al., 2020) . In this setting, the more reasoning steps, the longer the sequence the decoder must generate at once.

Step-by-step: The model outputs a single reasoning step per call. Each generated step is concatenated to the past input, and the model again generates the next step (i.e., *proofwriter* style) (Liang et al., 2021; Sanyal et al., 2022; Picco et al., 2021; Tafjord et al., 2021; Shwartz et al., 2020) . This process is iterated until the model outputs the answer or until a set maximum number of iterations is reached (100). **Token-by-token:** This is the same as step-be-step chaining, but the decoder outputs only a single *token* per call. We set the maximum number of steps to 500.

Comparing *all-at-once* and the others reveals the effect of changing the sequence length that the decoder outputs in a single call. In addition, comparing *step-by-step* and *token-by-token* quantifies the advantage of breaking a problem into meaningful units.

2.2 Chaining strategies

Particular variables sometimes depend on another variable; the key to reaching the correct answer is determining the order in which the equations are referred to. Regarding existing studies, we compared three chaining strategies: *shortest-path*, *exhaustive*, and *backward* chaining (Figure 2b).

Shortest-path chaining: The model straightforwardly solves the equations starting from the first solvable one (i.e., involving a known value) and ending with the target (Wei et al., 2022; Cobbe et al., 2021; Yavuz et al., 2022; Shwartz et al., 2020). Here, the reasoning behind determining the shortest path is not outputted by the model.

Exhaustive chaining: The model greedily solves all given equations until the target value is reached (Tafjord et al., 2021; Liang et al., 2021; Yang et al., 2022). Specifically, the model calculates the left-most solvable equation in each step. Note that this strategy typically derives a long reasoning chain; from an engineering perspective, this strategy is inefficient.

Backward chaining: The model starts from the equation for the target variable and backtracks over the dependent equations until it reaches a known value (Picco et al., 2021; Rocktäschel and Riedel, 2017; Cingillioglu and Russo, 2019). Then, it solves each equation in order by inserting known or calculated values until the target one is reached.

No chaining: As a baseline, we also examined the setting where the model was trained to directly output the answer.
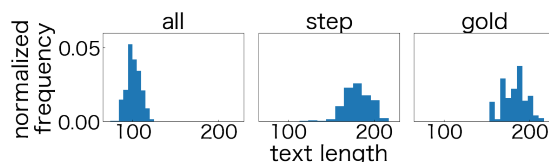

Figure 3: Distributions of the total reasoning chain length (num. characters). The all-at-once and step-by-step generate those at depth 12.

3 Results

Models: We used the pre-trained T5-base, T5-large [2] (Raffel et al., 2020), and BART-base [3] (Lewis et al., 2020). Results of BART-base are in Appendix C.

Note that their pre-defined tokenizers have all the numbers from 0 to 9, and the numerical values in our dataset are divided into digits (e.g., "12" should be "@@1 @@2") in advance, following Kim et al. (2021).

Training: The models were first pre-trained using a 10K *simple* dataset for 30 epochs, then trained with the 5K training set (1K training instances for each reasoning depth.) for 2000 epochs. The experiment setting details are in Appendix A. In addition, we prepared 0.2K test instances for each reasoning depth. This pre-training is intended to teach the models primitive operations (i.e., assignment, reference, and addition). The pre-training dataset contains two types of single-depth instances: *assign-refer* type (e.g., A=1,A?) and *operate-assign-refer* type (e.g., A=1+3, A?). All the results in the paper are averages of the results on three different seeds.

3.1 Output strategies

We compared the output strategies while fixing the chaining strategy to the shortest path. Figure 4a shows the accuracy per reasoning depth. Note that the accuracy score here denotes whether the answer (e.g., C=6) is correct. We observed the following: (i) **generating intermediate reasoning steps enhance the performance**, and (ii) among the output strategies, **step-by-step works the best**, and **all-at-once works the worst**. The format of the dataset in this study is simple. Therefore, this result indicates the low symbolic reasoning ability of neural models and the necessity of the choice of

---

[2]https://huggingface.co/docs/transformers/model_doc/T5
[3]https://huggingface.co/docs/transformers/model_doc/bart
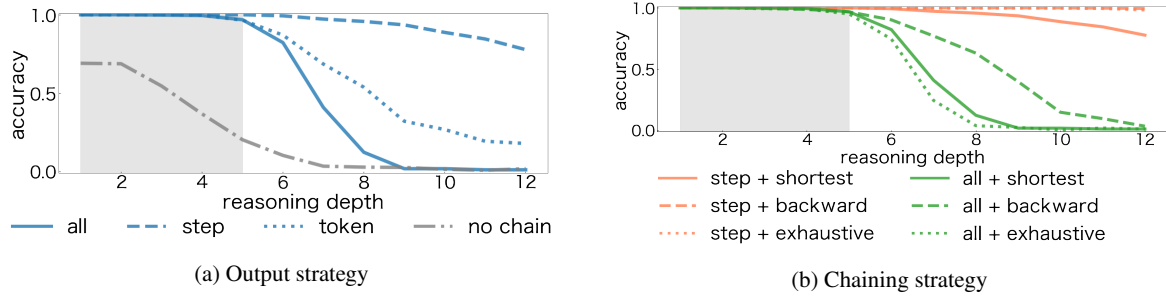
(a) Output strategy

(b) Chaining strategy

Figure 4: Accuracy changes of the models against reasoning depth. The gray range represents the training data domain (1-5 depth). Figure (4a) shows the performance degradation with the increase of reasoning steps when using the all-at-once strategy. Figure (4b) shows that the combination of step-by-step output and backward/exhaustive chaining leads to successful generalization.

Depth	Shortest	Backward	Exhaustive
6	99.3/99.3	100/ 100	99.7/99.7
8	95.5/95.7	100/ 100	99.8/99.8
12	76.7/77.7	99.5/99.5	98.2/98.3

Table 1: Accuracy of the T5-base model with the step-by-step output strategy at each depth (chain/answer).

Question: A=1, B=2+A, B?

Error types	Gold	Prediction
Copying error	**B=2+A**, B=2+1, B=3	**B=6+A**, B=6+1, B=7
Hasty assignment	**B=2+A**, B=2+1, B=3	(skip step) B=2+2, B=4

Table 2: Illustrative examples of the errors under the step-by-step, shortest-path chaining settings. (skip step) denotes that the reasoning steps is accidentally skipped.

an appropriate reasoning strategy.

We hypothesized that the source of all-at-once's inferiority was that the decoder overfitted to output a similar length of reasoning steps as those in the (shallower) training data. In fact, the models generated relatively shorter reasoning steps in the out-of-domain (e.g., depth of 12) setting when using the all-at-once strategy (Figure 3); this supports our hypothesis.

The advantage of step-by-step over token-by-token suggests the advantage of breaking the problem into meaningful units (reasoning step) and modeling each step in a single call of the encoder-decoder.

3.2 Chaining strategies

Figure 4b and Table 1 show the results on each depth with a fixed step-by-step output strategy.

Note that the accuracy of the chain (left side of the scores) was measured based on not an exact match but mathematically. For example, even if the order of generated equations is different, it is correct. The results of a fixed token-by-token output strategy are in Appendix B.

While the performance dropped in the shortest-path setting as the reasoning depth increased, with either the exhaustive or backward chaining, models successfully solved the task even when extrapolating to depths 6-12. The models correctly generated the intermediate steps (nearly perfect) as well as the final answer in the exhaustive and backward chaining settings (Table 1). Note that these strategies were ineffective with all-at-once outputting.

Gontier et al. (2020) compared chaining strategies and concluded that models that *didn't* generate reasoning steps had better generalization performance than models that did when the reasoning chains were long. However, our results suggest that the choice of the appropriate output strategy improves the reasoning ability of the model.

We considered that the source of shortest-path inferiority was the rough granularity of the given reasoning steps. The models don't know the shortest path before outputting the reasoning steps. Therefore, both the exhaustive and shortest path chaining approaches must search for variables other than those on the shortest path. As shown in Figure 2b, the exhaustive chaining approach is taught this process explicitly. On the other hand, the shortest-path chaining approach must be learned that by training data that don't include this process. We thought this difference affected the accuracy and concluded that **the accuracy is higher when the granularity of given intermediate steps is**

finer, even though they are long.

Therefore, we concluded that **the accuracy is higher when the granularity of intermediate steps is finer**, even though they are long.

3.3 Error analysis

We also analyzed the errors of the depth-12 instances under the shortest-path strategy. [4] We observed two types of errors: (i) copying errors and (ii) hasty assignment. Table 2 shows an illustrative example of each error type and the percentage of these errors. The most frequent one (53%) was a simple copying error, where the model failed to accurately copy an original equation into the reasoning chain. This erroneous copying ability is consistent with Xu et al. (2020) and supports the advantage of introducing a copy mechanism to the model (Ontanon et al., 2022). Second, a hasty assignment is the model skipping the step to copy the equation from context and instead assigned it a random value. Note that these errors were almost addressed in the other strategies; this could stem from the difficulty of the implicit calculation of the shortest path.

3.4 Models' scalability

To investigate the scalability, we compared T5-large with T5-base. Figure 5 shows the result. T5-large had a similar trend but slightly lower accuracy on all-at-once and step-by-step compared to T5-base. The reason may be that T5-large needs more data for updating the weights of the entire model. On the other hand, the accuracy of T5-large is higher than T5-base on token-by-token. It's because the data size of token-by-token is as token lengths of output sequence times as the data size of all-at-once, as shown in Figure 2a. This result indicates that the parameter size of the model needs to be larger to output token-by-token.

4 Conclusions

We investigated and factorized the reasoning strategy in symbolic numerical reasoning with neural seq2seq models. We found that the combination of step-by-step output and finely granular reasoning leads to successfully performing symbolic reasoning. Our results support the potential of neural models for symbolic reasoning.

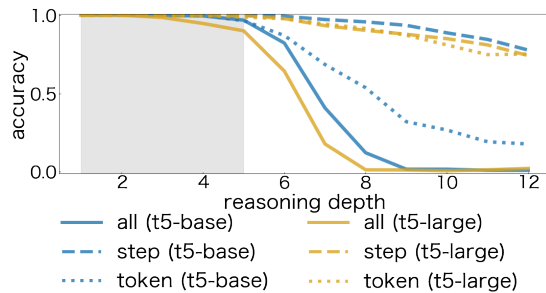[4]In total, 32 instances were analyzed. That is the total number of incorrect answers on one seed.

Figure 5: Accuracy changes of the T5-base and T5-large against reasoning depth. The gray range presents the training data domain (1-5 depth). This figure shows that the accuracy of T5-large with token-by-token is higher.

Limitations

We found that even simple symbolic reasoning requires the appropriate selection of reasoning strategy. It is unclear whether our findings generalize to more complex symbolic reasoning and/or problems written in natural language. If our findings do not generalize in these different settings, we must address the gap in future work. For example, we start with one of the simplest tasks and find out when models fail as we add complexity to tasks one by one.

From the engineering perspective, the iterative strategies are limited to the input length of the model. For example, in our experiments, when adopting the setting where reasoning depths are greater than 13, the input length of step-by-step and token-by-token became longer than the input length limit of T5 (i.e., 512 tokens).

In addition, gigantic language models (e.g., GPT-3) have recently been used. Including these models in our study is one of our future works.

References

Hadeel Al-Negheimish, Pranava Madhyastha, and Alessandra Russo. 2021. Numerical reasoning in machine reading comprehension tasks: are we there yet? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021,*

pages 9643–9649. Association for Computational Linguistics.

Nuri Cingillioglu and Alessandra Russo. 2019. Deeplogic: Towards end-to-end differentiable logical reasoning. In *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019., Stanford University, Palo Alto, California, USA, March 25-27, 2019*, volume 2350 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168.

Artur d'Avila Garcez and Luís C. Lamb. 2020. Neurosymbolic AI: the 3rd wave. *CoRR*, abs/2012.05876.

Nicolas Gontier, Koustuv Sinha, Siva Reddy, and Christopher Pal. 2020. Measuring systematic generalization in neural proof generation with transformers. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ashim Gupta, Giorgi Kvernadze, and Vivek Srikumar. 2021. BERT & family eat word salad: Experiments with text understanding. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12946–12954. AAAI Press.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R. Bowman, and Noah A. Smith. 2018. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 107–112. Association for Computational Linguistics.

Kyle Hamilton, Aparna Nayak, Bojan Bozic, and Luca Longo. 2022. Is neuro-symbolic AI meeting its promise in natural language processing? A structured review. *CoRR*, abs/2202.12205.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 2021–2031. Association for Computational Linguistics.

Jeonghwan Kim, Giwon Hong, Kyung-min Kim, Junmo Kang, and Sung-Hyon Myaeng. 2021. Have you seen that number? investigating extrapolation in question answering models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 7031–7037. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *CoRR*, abs/2205.11916.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *CoRR*, abs/2206.14858.

Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. 2021. Explainable multi-hop verbal reasoning through internal monologue. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1225–1250. Association for Computational Linguistics.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3428–3448. Association for Computational Linguistics.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114.

Santiago Ontanon, Joshua Ainslie, Zachary Fisher, and Vaclav Cvicek. 2022. Making transformers solve compositional tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers), pages 3591–3607, Dublin, Ireland. Association for Computational Linguistics.

Gabriele Picco, Thanh Lam Hoang, Marco Luca Sbodio, and Vanessa López. 2021. Neural unification for logic reasoning over natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3939–3950. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Gabriel Recchia. 2021. Teaching autoregressive language models complex tasks by demonstration. *CoRR*, abs/2109.02102.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3788–3800.

Soumya Sanyal, Harman Singh, and Xiang Ren. 2022. Fairr: Faithful and robust deductive reasoning over natural language. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1075–1093. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4615–4629. Association for Computational Linguistics.

Saku Sugawara, Kentaro Inui, Satoshi Sekine, and Akiko Aizawa. 2018. What makes reading comprehension questions easier? In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 4208–4219. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3621–3634. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Song Xu, Haoran Li, Peng Yuan, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Self-attention guided copy mechanism for abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1355–1362. Association for Computational Linguistics.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. *CoRR*, abs/2205.12443.

Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, Nitish Shirish Keskar, and Caiming Xiong. 2022. Modeling multi-hop question answering as single sequence prediction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 974–990. Association for Computational Linguistics.
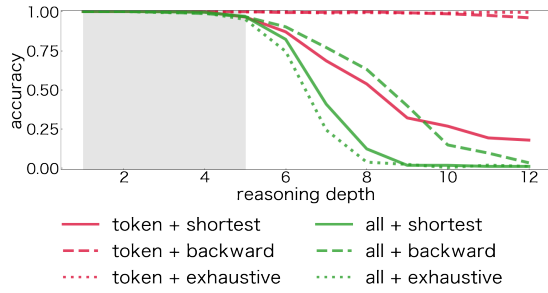
Figure 6: Accuracy changes of token-by-token per reasoning depth. The gray range presents the training data domain (depths 1-5).
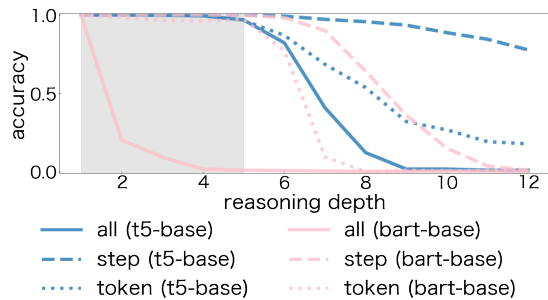

Figure 7: Accuracy changes of the T5-base and BART-base models per reasoning depth. The gray range presents the training data domain (depths 1-5). T5 seems to outperform BART.

A Details on Experimental Settings

We first examined the learning rate from 10^{-3}, 10^{-4}, and 10^{-5}; among them, we used the largest rate at which the loss converged. After training models, we used the model with the lowest validation loss among the per-epoch checkpoints during the training reported. We used four NVIDIA V100 GPUs for NVLink 16GiB HBM2.

B Results of Token-by-token

Figure 6 shows the results on each depth with a fixed token-by-token output strategy. Like step-by-step, the performance drops in the shortest-path setting as the reasoning depth increases. In addition, the exhaustive or backward successfully solves the task even when extrapolating to depths 6-12.

C Different Architectures

We also tested BART-base (Lewis et al., 2020) as a baseline to investigate the effectiveness of the NLP-task-oriented objectives used in the T5-style pre-training. Figure 7 shows this result. In this particular setting, T5 was superior to BART. This suggests that the NLP-task-oriented objectives benefit symbolic reasoning.

D Other errors

We analyzed the cases where the answer is correct and the chain is wrong. Table 3 shows examples of chain errors. Ignoring the incorrect step is an example of the model outputting the correct reasoning step after outputting an incorrect one. Correct assignment is an example in which the assignment accidentally makes the model output the correct step. Finally, Non-affecting error is an example in which a variable not on the shortest path is wrongly assigned a value.

Question: A=1, C=5+B, B=2+A, D=3+A, C?

Chain error types	Gold	Prediction
Ignoring the incorrect step	A=1, **B=2+A**, B=2+1, B=3, C=5+B, C=5+3, C=8	A=1, **B=2+D**, **B=2+A**, B=2+1, B=3, C=5+B, C=5+3, C=8
Correct assignment	A=1, **B=2+A**, **B=2+1**, B=3, C=5+B, C=5+3, C=8	A=1, **B=2+D**, **B=2+1**, B=3, C=5+B, C=5+3, C=8
Non affecting error	A=1, B=2+A, B=2+1, B=3, C=5+B, C=5+3, C=8	A=1, B=2+A, B=2+1, B=3, **D=3+A, D=3+2, D=5**, C=5+B, C=5+3, C=8

Table 3: These instances are examples of chain errors. Note that the final answers are correct.