A. Synthetic Task Generation Pseudocode

Algorithm 1

function GENERATE_TUPLE(Vocabulary size S)
  Vocabulary $\mathcal{V} \leftarrow \{1, 2, \ldots, S\}$. ▷ Use an integer representation of symbols.
  Math symbol set $\mathcal{M} \leftarrow$ SAMPLE($\mathcal{V}$, n=44, replacement=False). ▷ Sample 44 distinct symbols.
  Rule symbol set $\mathcal{R} \leftarrow$ SAMPLE($\mathcal{V} \backslash \mathcal{M}$, n=20, replacement=False). ▷ Sample 20 distinct symbols.
  Rule $R \leftarrow$ SAMPLE($\mathcal{M} \bigcup \mathcal{R}$, n=RANDOM(5,20), replacement=False). ▷ Sample a sequence of symbols of length between 5 and 20.
  Case dictionary $C \leftarrow \{\}$.
  for s in $\mathcal{R}$ **do**
    Case dictionary $C[s] \leftarrow$ SAMPLE($\mathcal{M}$, n=RANDOM(2,8), replacement=True). ▷ Sample a sequence of symbols for each rule symbol, of length of length between 2 and 8.
  end for
  Result $R' \leftarrow$ Rule R. ▷ Set result string R' to be the same as rule string R.
  for s in $\mathcal{R}$ **do**
    SUBSTITUTE(R', s, $C[s]$). ▷ Substitute every rule symbol s in result string R' with previously randomly sampled string $C[s]$.
  end for
  return Math symbol set $\mathcal{M}$, Rule symbol set $\mathcal{R}$, Rule R, Case C, Result R'.
end function

B. Other synthetic tasks

In this section, we give descriptions of other variants of the synthetic tasks we considered than the ones introduced in the main paper.

B.1. `Rewrite` and `Rewrite_multistep`

We propose a rewrite task, inspired by the rewrite tactic used in interactive theorem provers. The `Rewrite` task requires the model to rewrite a string according to a rule transformation. One example of the task is:

Source: $a + b - c$ <s> $A + B = B + A$

Target: $b + a - c$

"$A + B = B + A$" is the rule transformation, which is applied to the LHS string "$a + b - c$". The model needs to predict the RHS string as the result of the rule application, i.e., $b + a - c$. Besides rule symbols and math symbols, we also require the third set of symbols, named as "string symbols". For the ease of our discussion, we we will think of math symbols as the union of those operators used in mathematics (e.g., "$+ - * = ()\&$"), rule symbols as upper case letters (e.g., A, B, $C \ldots$), and string symbols as lower case letters (e.g., a, b, $c \ldots$). We first sample a random string as the LHS string, consisting of math symbols and string symbols (e.g., $a + b - c$). We sample a sub-string of the LHS string, and replace the string symbols in the sub-string with rule symbols. For example, we sample and obtain the substring $a + b$ from $a + b - c$, and we replace a, b with rule symbols A, B. This then forms the LHS of the rule transformation, $A + B$, with the substitution dictionary $\{A : a, B : b\}$. We then sample the RHS of the rule transformation from the union of rule symbols A and B, and all math symbols, e.g., $B + A$. This gives the rule transformation $A + B = B + A$. We substitute the value of the substitution dictionary for each rule symbol in the RHS rule, and then substitute back to the original LHS string to obtain $b + a - c$. The task example is constructed by using the LHS string and the rule transformation as the source input, and use the result of the rule transformation as the target.

We further introduce a multi-step version of the rewrite task: `Rewrite_multistep`. In this task, the source may contain more than one rewrite rule, and the target is the result of applying all the rewrite rules in a sequence. This task is motivated from the need to perform multi-step planning in mathematical reasoning tasks. During pre-training, for each training example, we uniformly sample the number of rewrite steps from 1 to 5.

Table 9. Test top-1, top-10 (%) accuracy on the IsarStep task.

Model	Top-1 Acc.	Top-10 Acc.
No pretrain (Li et al., 2021)	20.4	33.1
HAT (Li et al., 2021)	22.8	35.2
LIME `Deduct`	24.7	37.7
LIME `Abduct`	26.7	41.0
LIME `Induct`	23.9	38.8
LIME `Mix`	26.9	40.4
LIME `Rewrite`	26.0	38.6
LIME `Rewrite_multistep`	**28.6**	**43.9**
LIME `Induct_v2`	25.6	39.8
LIME `Induct_v3`	25.0	38.8
LIME `Induct_rewrite`	25.8	39.5

Table 10. Vocabulary sizes' effects on the IsarStep task.

Model	Top-1 Acc.	Top-10 Acc.
No pretrain	20.4	33.1
LIME on `Rewrite`, $S = 100$	24.1	37.5
LIME on `Rewrite`, $S = 512$	25.4	38.8
LIME on `Rewrite`, $S = 1000$	26.0	38.6
LIME on `Rewrite`, $S = 5000$	25.8	38.5
LIME on `Rewrite`, $S = 25000$	27.4	40.9

B.2. Other variants of `Induct` Task

We introduce three other variants of the `Induct` task.

1. `Induct_v2`: We move the Case dictionary from the source input to the target output. This makes the task significantly harder, which requires the agent to synthesize a rule and a possible explanation (Case) to explain the Result.
2. `Induct_v3`: Instead of providing the Case dictionary, we provide two Result strings, coming from the same Rule. Namely, we sample two Case dictionaries, and applying each to the Rule string to obtain two Result strings. Both Result strings are used as source, and the target is the Rule string.
3. `Induct_rewrite`: We also create a "induction" version of the `Rewrite` task. In this task, the source is the LHS string concatenated with the RHS string, that is the result of the rewrite. The target is the rewrite rule that is used to do the rewrite.

B.3. A full comparison of all synthetic tasks

In this section we present a full comparison for all synthetic tasks. We followed the training protocol in 4.1 and evaluate the method on IsarStep. The results are reported in Table 9. We can see that the `Rewrite_multistep` achieved the best performance across all synthetic tasks, surpassing the baseline by 8.2% for Top-1 accuracy and 10.8% for Top-10 accuracy. This indicates the inductive bias for long horizon reasoning encoded in `Rewrite_multistep` is very useful for the reasoning task.

C. More Ablation Studies

C.1. Does the vocabulary size matter?

In this section, we investigate whether the vocabulary size S in the synthetic task generation algorithm has an effect on the performance. We used the `REWRITE` task for the experiment in this section. We generated datasets of various vocabulary sizes, 100, 512, 1000, 5000, 25000. We used the same curriculum learning for pre-training as described in 4.1 on larger vocabulary sizes: first training on the `Rewrite` task of vocabulary size 100 for 10K steps, then training on each individual

dataset for another 10K steps. We compare the performance on the downstream task Isarstep. The results are presented in Table 10. We see that when the vocabulary size is equal or larger than 512, the performance were similar. The smallest vocabulary size 100 obtained the worst performance among all, and all the other four models achieved similar BLEU scores. The model trained on the largest vocabulary achieved best performance on top-1 accuracy and top-10 accuracy. The results show there is a non-trivial effect of the vocabulary size of the synthetic task to the performance of the downstream task. Hence we use vocabulary size of 1000 for all the experiments in the main paper. We leave investigations of the causes to future work.