A Detailed Statistics of Datasets

We show the statistics of all the datasets used in our paper in Table 8.

For CLUTRR, we follow the setting in FAITHFULCOT [27] and construct the prompt using exemplars requiring 2-3 reasoning steps and test whether the model can generalize to examples requiring up to 10 steps. We used the pre-processed test data consisting of 1,042 test examples from past work [27].

For PROOFWRITER, we use the closed world assumption setting, following past work [8]. We construct our test set by randomly sampling a subset of 1,000 examples (out of 10,000) from the test split of depth-5 setting, the most challenging setting.

For STREGEX, we merge the test and test-E split (see original paper [50]) to form a test set consisting of 996 examples in total.

Table 8: Number of few-shot exemplars, number of test examples and license for the datasets used in our paper.

	# Shot	# Test	License
GSM [7]	8	1,319	MIT license
GSM-SYS	8	547	MIT license
ALGEBRA [18]	8	222	Creative Commons Attribution Share Alike 4.0
LSAT [56]	8	230	MIT license
CLUTRR [42]	8	1,042	Attribution-NonCommercial 4.0
PROOFWRITER [43]	4	1,000	CC BY 4.0.
COLOREDOBJECT (BIG-BENCH)	3	2,000	Apache 2.0
STRUCTUREDREGEX [50]	8	996	MIT license

GSM-SYS Dataset We construct GSM-SYS, a special subset consisting of 547 examples extracted from GSM. Specifically, we filter the entire GSM dataset (train split + test split) to find examples whose human-annotated explanations involve a system of equations, using patterns like *"let [letter] be"*, *"assume [letter] be"* and *"[number][letter]"*. We manually inspected 10% of the examples and found 80% of those samples did involve systems of equations in the explanation. We refer to this more challenging dataset as GSM-SYS.

B Details of the Prompts

In general, we leverage COT prompts and PROGLM prompts from existing work whenever available, and manually write SATLM prompts for the **same exemplar sets**. Prompt examples for all datasets can be found in Appendix G.

For **GSM and GSM-SYS**, we adapt the original COT prompt and PROGLM prompt used in program-aided language models [14]. Specifically, we replace one random exemplar in the original prompt with another exemplar sampled from GSM-SYS. This is to improve the performance of COT and PROGLM on GSM-SYS, as the original exemplar set achieves suboptimal performance for GSM-SYS. Our adapted COT and PROGLM prompts achieve better performance compared to the original ones on both GSM and GSM-SYS (see Appendix C for details).

For **LSAT**, we randomly sample 8 exemplars and write prompts for COT and SATLM. We note that LSAT is a particularly challenging task: we tried 3 CoT prompts written by 3 different authors of our paper, which all led to around 20% accuracy. Similar results are reported in other work [24, 38]. In addition, we only report COT results, leaving out PROGLM. This decision is due to the fact that PROGLM uses Python as its program interpreter. While Python is a general-purpose programming language, it does not provide native support for formal logic reasoning, including essential components like logical inference rules and manipulation of logical formulas. Solving problems from LSAT requires strategies like proof by contradiction (see Appendix G for a detailed example), which we see no way to represent in the PROGLM framework and is not addressed in prior work.

For **CLUTRR**, we use the COT prompt and PROGLM prompt provided in FAITHFULCOT [27]. For **PROOFWRITER**, we use the COT prompt from SELECTION-INFERENCE [8], and adapt it to form the ProgLM prompt. We use the CoT prompt and ProgLM from Pal [14] for **Colored Object**.

The task of **StructuredRegex**, a regex synthesis dataset, is to parse natural language descriptions to regexes. This is not a typical reasoning dataset, and there is no CoT prompt for this dataset. We randomly sample 8 exemplars and annotate the prompt for ProgLM and SatLM. In this setting, ProgLM directly translates NL descriptions into regexes (which are essentially programs), whereas SatLM parses an NL description into a set of constraints over the surface form of the regex. Note that this dataset provides *multimodal* specifications of regexes, featuring both NL descriptions and examples. The I/O examples can be used to reject synthesized regexes if they do not accept or reject the correct examples. When we report results for self-consistency inference, we follow past work [51] and filter out incorrect outputs using the I/O examples provided in the dataset [50]. (This setting therefore checks consistency with something other than the model itself, but uses a similar computation budget as self-consistency, so we group it with those results.)

C Performance of Original CoT and ProgLM Prompts on Arithmetic Reasoning Datasets

Table 9: Performance of different approaches using our adapted exemplar set and the original exemplar set used in CoT and Pal.

	Adapted (Ours)		Original	
	GSM-Sys	GSM	GSM-Sys	GSM
CoT	46.5	62.7	35.7	62.4
ProgLM	43.4	**72.7**	36.1	**71.7**
SatLM	**69.4**	71.8	**66.7**	70.9

Recall that we construct our arithmetic reasoning prompt used in Table 1 by replacing one random exemplar in the original prompt used in CoT and ProgLM with an random example from GSM-Sys. We show the performance of CoT, ProgLM, and our SatLM in Table 9 using our adapted exemplar set and original exemplar set in Table 9.

Our adaptation significantly improves the performance of CoT and ProgLM on GSM-Sys, and slightly improves the performance on GSM. Furthermore, we still see that SatLM outperforms both CoT and ProgLM by a large margin on GSM, using either our adapted set or the original set.

D Extended Discussion on Concurrent Work

Table 10: Performance of different approaches on Algebra.

	Algebra	GSM
CoT	53.6	62.4
ProgLM	55.7	72.7
SatLM (Ours)	77.9	71.8
MathSym [18]	76.3	69.4

Similar to our work, [18] proposes to solve arithmetic reasoning problems by parsing the problem into a set of variables and equations and using an external solver to derive the final answer. While their formalization is restricted to arithmetic problems, we use SAT problems encoded with first-order logical formulas that unifies a wide range of reasoning tasks.

In addition, we also evaluate our approach on the Algebra dataset in [18], which consists of 222 examples from Algebra textbooks. We note that the results between ours and MathSym are not directly comparable, as MathSym picks a different exemplar set. As shown in Table 10, Algebra is more challenging than GSM, and SatLM outperforms ProgLM and CoT by more than 20%.

E Examples of Error Cases of SatLM

E.1 Examples of Unsatisfiable and Ambiguous Formulas

We show examples of formulas that are unsatisfiable or have ambiguous answers in Figure 4, Figure 5, and Figure 6.

UNSAT Formulas: GSM

Q: If a rectangle has a width of 42 inches and an area of 1638, how many rectangles of the same size would reach a length of 390 inches?

```
rectangle_width = 42
rectangle_area = 1638
rectangle_length = 390
rectangle_length = rectangle_area / rectangle_width
rectangles_needed = rectangle_length / rectangle_width
result = rectangles_needed
solve(result)
```

Q: Jason's dog has a tail that's half the length of its body, and a head that's 1/6 the length of its body. If the dog is 30 inches long overall, how long is its tail?

```
dog_length = 30
tail_length = Variable()
head_length = Variable()
dog_length = tail_length + head_length
tail_length = dog_length * 1 / 2
head_length = dog_length * 1 / 6
result = tail_length
solver(result)
```

Figure 4: Examples of UNSAT formulas from GSM. Both examples are generated by `code-davinci-002`. UNSAT occurs when LLMs misunderstand the problem and output conflicting constraints (highlighted in red).

AMBIG Formulas: GSM

Q: The sum of the three numbers is 500. If the first number is 200, and the value of the second number is twice the value of the third number, find the value of the third number.

```
num1 = 200
num2 = Variable()
num3 = Variable()
num2 = num3 * 2
total = num1 + num2 + num3
result = num3
solve(result)
```

Q: The total for the sum and product of Elvie's age and Arielle's age are 131. If Elvie's age is 10, how old is Arielle?

```
elvie_age = 10
arielle_age = Variable()
sum_age = elvie_age + arielle_age
product_age = elvie_age * arielle_age
total_age = sum_age + product_age
result = arielle_age
solve(result)
```

Figure 5: Examples of AMBIG solutions from GSM. Both examples are generated by `code-davinci-002`. The ambiguous formulas are under-constrained due to failure in encoding certain constraints specified in the problem (highlighted in red), resulting in multiple possible answers.

UNSAT Formulas: CLUTRR

Arlene and her husband Jeff went on a cruise. They had a wonderful time. Stephanie s father Jason loves his little princess even though she gets into a lot of trouble at school. Gloria's mother Ruth and brother Jeff were working in the kitchen, preparing soup. Stephanie, Ruth's daughter, was working outside in the garden.
Q: How is [Jason] related to [Arlene]?

```
# [Arlene] and her husband [Jeff] went on a cruise. They had a wonderful time.
relation(Arlene, Jeff) = (wife, husband)
# [Stephanie]'s father [Jason] loves his little princess even though she gets into a lot of trouble at school.
relation(Stephanie, Jason) = (daughter, father)
# [Gloria]'s mother [Ruth] and brother [Jeff] were working in the kitchen, preparing soup.
relation(Gloria, Ruth) = (daughter, mother)
relation(Gloria, Jeff) = (daughter, brother)
# [Stephanie], [Ruth]'s daughter, was working outside in the garden.
relation(Stephanie, Ruth) = (daughter, mother)
# How is [Jason] related to [Arlene]?
solve(relation(Jason, Arlene))
```

AMBIG Formulas: CLUTRR

Kirk loves talking to his grandfather Stanley on the phone Paul's brother, Kirk, recently made it into town. Paul and his son Michael went to look at cars. Paul ended up buying the Mustang.
Q: How is [Stanley] related to [Michael]?

```
# [Kirk] loves talking to his grandfather [Stanley] on the phone.
relation(Kirk, Stanley) = (grandson, grandfather)
# [Paul]s brother, [Kirk], recently made it into town.
relation(Paul, Kirk) = (brother, brother)
# [Paul] and his son [Michael] went to look at cars.
relation(Paul, Michael) = (father, son) note: Michael is Kirk's son, not Paul's son
# How is [Stanley] related to [Michael]?
solve(relation(Stanley, Michael))
```

Figure 6: An example of UNSAT formulas (top) and an example of AMBIG formulas (bottom) from CLUTRR. Both happen when the LLM misparses a constraint (highlighted in red). In particular, ambiguous formulas usually occur when the language description itself is highly ambiguous (see the note in the ambiguous formulas for an example).

E.2 Examples of Incorrect Answers

We show examples where the SAT solver successfully returns an answer, but the answer is incorrect in Figure 7.

Examples of Incorrect Answers: GSM

Q: Martin rings the small bell 4 times more than 1/3 as often as the big bell. If he rings both of them a combined total of 52 times, how many times does he ring the big bell?

```
small_bell_rings = Variable()
big_bell_rings = Variable()
total_bell_rings = 52
small_bell_rings = big_bell_rings * 1 / 3 * 4
total_bell_rings = small_bell_rings + big_bell_rings
result = big_bell_rings
solve(result)
```

Q: Jonathan was sad to learn he needed 2 more toys than he already had to have enough toys to make a sorted play set five times larger than James's play set, which had 80 toys. How many toys does Jonathan currently have?

```
james_toys = 80
jonathan_toys = Variable()
jonathan_toys_needed = james_toys * 5 + 2
result = jonathan_toys_needed - 2
solve(result)
```

Examples of Incorrect Answers: CLUTRR

Donald and Russell are brothers. Jason is father of their father.
Q: How is [Russell] related to [Jason]?

```
# [Donald] and [Russell] are brothers.
relation(Donald, Russell) = (brother, brother)
# [Jason] is father of their father
relation(Jason, Donald) = (father, son)
relation(Jason, Russell) = (father, son)
# How is [Russell] related to [Jason]?
solve(relation(Russell, Jason))
```

Figure 7: Two examples of incorrect answers from GSM (top) and an example of incorrect answer from CLUTRR (bottom). This occurs when LLMs output incorrect constraints (highlighted in red) but the incorrect formulas can still derive a non-ambiguous answer.

F Examples of Execution Errors and Planning Errors Made by CoTSolver

We show examples of execution errors and planning errors produced by CoTSolver in Figure 8, Figure 9, and Figure 10.

As shown in Figure 8, LLMs can make errors when executing arithmetic operators, resulting in execution errors. While introducing an interpreter (ProgLM) can help fix such execution errors, it is not effective for handling planning errors in Figure 9, which involves manipulating symbolic variables to determine a chain of arithmetic equations leading to the final answer.

We note that for CLUTRR, the atomic function is applying a set of transitivity rules that infer the relation between `A` and `C` based on the relation between `A` and `B` and the relation between `B` and `C` (e.g., `A's son is B, B's daughter is C. So A's granddaughter is C`). So the an execution error in CLUTRR refers to incorrect outputs when applying transitivity rules (see the top section of Figure 10). The set of transitivity rules we use are provided in [27, 54].

Execution Errors of CoTSolver: GSM

Q: Carly is thinking about buying a wallet that costs 22 and a purse that costs 3 less than four times the cost of the wallet. What is the combined cost of both items?

```
wallet_cost = 22
purse_cost = Variable()
purse_cost = wallet_cost * 4 - 3
total_cost = wallet_cost + purse_cost
result = total_cost
# We know purse_cost = wallet_cost * 4 - 3
# So total_cost = wallet_cost + purse_cost = 22 + (22 * 4 - 3) = 22 + 87 = 109
# result = total_cost = 109
# The answer is 109
```

Q: A person borrowed an amount of money for a year at an interest rate of 12%. If the total interest is $1500, what was the original borrowed amount?

```
interest_rate = 12
interest_total = 1500
borrowed_amount = Variable()
interest_total = borrowed_amount * interest_rate / 100 result = borrowed_amount
# We know interest_total = borrowed_amount * interest_rate / 100
# So borrowed_amount = interest_total / interest_rate * 100 = 1500 / 12 * 100 = 12500
/ 12 = 1041.67
# result = borrowed_amount = 1041.67
# The answer is 1041.67
```

Figure 8: Examples of execution errors made by CoTSolver on arithmetic reasoning tasks. Both examples are generated by `code-davinci-002`, which makes mistakes when calculating the final value of equations.

Planning Errors of CoTSolver: GSM

Q: Griffin had 24 french fries, but Kyle took 5 of them. Billy took twice as many as Kyle. Ginger gave Griffin a handful of her fries, and then Colby took from Griffin 3 less than the number of fries that Kyle had taken. If in the end Griffin had 27 fries, how many fries did Ginger give Griffin?

```
fries_initial = 24
kyle_took = 5
billy_took = kyle_took * 2
ginger_gave = Variable()
colby_took = kyle_took - 3
fries_after = 27
fries_after = fries_initial - kyle_took - billy_took + ginger_gave - colby_took
result = ginger_gave
# Solve the value of result
# We know fries_after = fries_initial - kyle_took - billy_took + ginger_gave - colby_took
# So ginger_gave = fries_after - fries_initial + kyle_took + billy_took - colby_took =
27 - 24 + 5 + 10 - 2 = 16
# result = ginger_gave = 16
# The answer is 16
```

Q: The combined age of Peter, Paul and Jean is 100 years old. Find the age of Peter knowing that Paul is 10 years older than John and that Peter's age is equal to the sum of Paul and John's age.

```
john_age = Variable()
total_age = 100
paul_age = john_age + 10
peter_age = paul_age + john_age
total_age = peter_age + paul_age + john_age
result = peter_age
# We know paul_age = john_age + 10, and peter_age = paul_age + john_age. So peter_age
= john_age + 10 + john_age = 2 * john_age + 10
# We also know total_age = peter_age + paul_age + john_age = 2 * john_age + 10 + john_age
+ 10 = 3 * john_age + 20
# Therefore, john_age = (total_age - 20) / 3 = (100 - 20) / 3 = 80 / 3 = 26.67
# peter_age = 2 * john_age + 10 = 2 * 26.67 + 10 = 53.33 + 10 = 63.33
# result = peter_age = 63.33
# The answer is 63.33
```

Figure 9: Examples of planning errors made by CoTSolver from GSM. Both examples are generated by `code-davinci-002`. In the first example, LLM fails to flip the sign of `colby_took`. In the second example, the model drops an occurrence of `john_age` when expanding out the formula. In both case, LLM fails to arrange the symbols to derive a chain of numeric equations that gives the correct answer.

Execution Errors of COTSOLVER: CLUTRR

```
# [James] was looking forward to dinner with his granddaughter, [Joanne].
relation(James, Joanne) = (grandfather, granddaughter)
# [Helen] went to her son [Leo]'s House
relation(Helen, Leo) = (mother, son)
# [Gene] bought gifts for his lovely sister, [Joanne]
relation(Gene, Joanne) = (brother, sister)
# [Kecia] wanted to visit an art museum, so she asked her father, [James] to take her.
relation(Kecia, James) = (daughter, father)
# [Sharon] and her sister, [Kecia], frequently bicker. However, they always make up quickly.
relation(Sharon, Kecia) = (sister, sister)
# [Leo] was so happy for his brother, [Gene], getting a promotion.
relation(Leo, Gene) = (brother, brother)
# How is [Sharon] related to [Helen]?
answer = relation(Sharon, Helen)
# Solve the problem
# Helen's son is Leo, and Leo's brother is Gene, so Helen's son is Gene.
# Helen's son is Gene, and Gene's sister is Joanne, so Helen's daughter is Joanne.
# Helen's daughter is Joanne, and Joanne's father is James, so Helen's son is James.
# Helen's son is James, and James's daughter is Kecia, so Helen's granddaughter is Kecia.
# Helen's granddaughter is Kecia, and Kecia's sister is Sharon, so Helen's granddaughter is Sharon.
# The answer is granddaughter.
```

Planning Errors of COTSOLVER: CLUTRR

```
# [Rosa] took her daughter [Melissa] to the park.
relation(Rosa, Melissa) = (mother, daughter)
# [Melissa] was sad her sister [Annie] could not join.
relation(Melissa, Annie) = (sister, sister)
# [Lisa] is having a great day at the playground with her mother [Rosa] and [Rosa]'s sister [Ruth].
relation(Lisa, Rosa) = (daughter, mother)
relation(Rosa, Ruth) = (sister, sister)
# [Rosa] asked her daughter, [Lisa], if she would like to go to a movie with her on Saturday night.
relation(Rosa, Lisa) = (mother, daughter)
# How is [Ruth] related to [Annie]?
answer = relation(Ruth, Annie)
# Solve the problem
# Rosa's daughter is Lisa, and Lisa's mother is Rosa, so Rosa's mother is Rosa.
# Rosa's mother is Rosa, and Rosa's sister is Ruth, so Rosa's aunt is Ruth.
# Rosa's aunt is Ruth, and Rosa's daughter is Melissa, so Rosa's niece is Melissa.
# Rosa's niece is Melissa, and Melissa's sister is Annie, so Rosa's niece is Annie.
# The answer is niece.
```

Figure 10: Examples of planning errors made by COTSOLVER on CLUTRR. We omit questions for brevity. Both examples are generated by `code-davinci-002`. In the first example, the model outputs an incorrect value when applying the transitivity rule marked in red (correct output should be `husband`). In the second example, the model comes up with an incorrect procedure.

G Prompt Examples

We show one or two exemplars in the prompt for each dataset. We list prompts for PROGLM for comparison.

Prompts for GSM and GSM-SYS

SATLM

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?

```
jason_lollipops_initial = 20
lollipops_given = Variable()
jason_lollipops_after = 12
jason_lollipops_after = jason_lollipops_initial - lollipops_given
result = lollipops_given
solve(result)
```

Q: Jeff bought 6 pairs of shoes and 4 jerseys for $560. Jerseys cost 1/4 price of one pair of shoes. Find the shoe's price total price.

```
shoes_num = 6
jerseys_num = 4
total_cost = 560
shoes_cost_each = Variable()
jerseys_cost_each = Variable()
shoes_cost_each * shoes_num + jerseys_cost_each * jerseys_num = total_cost
jerseys_cost_each = shoes_cost_each * 1 / 4
shoes_cost_total = shoes_cost_each * shoes_num
result = shoes_cost_total
solve(result)
```

PROGLM from [14]

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?

```
jason_lollipops_initial = 20
jason_lollipops_after = 12
denny_lollipops = jason_lollipops_initial - jason_lollipops_after
result = denny_lollipops
return result
```

Q: Jeff bought 6 pairs of shoes and 4 jerseys for $560. Jerseys cost 1/4 price of one pair of shoes. Find the shoe's price total price.

```
shoes_num = 6
jerseys_num = 4
total_cost = 560
jersey_shoes_cost_ratio = 1 / 4
shoes_cost_each = total_cost / (shoes_num + jerseys_num * jersey_shoes_cost_ratio)
shoes_cost_total = shoes_cost_each * shoes_num
result = shoes_cost_total
return result
```

Figure 11: Prompt (excerpt) used for GSM and GSM-SYS.

Prompts for LSAT

SatLM

Nine different treatments are available for a certain illness: three antibiotics—F, G, and H—three dietary regimens—M, N, and O—and three physical therapies—U, V, and W. For each case of the illness, a doctor will prescribe exactly five of the treatments, in accordance with the following conditions: If two of the antibiotics are prescribed, the remaining antibiotic cannot be prescribed. There must be exactly one dietary regimen prescribed. If O is not prescribed, F cannot be prescribed. If W is prescribed, F cannot be prescribed. G cannot be prescribed if both N and U are prescribed. V cannot be prescribed unless both H and M are prescribed.
Question: If O is prescribed for a given case, which one of the following is a pair of treatments both of which must also be prescribed for that case?
(A) F, M (B) G, V (C) N, U (D) U, V (E) U, W

```
treatments = [F, G, H, M, N, O, U, V, W]
antibiotics = [F, G, H]
dietary_regimens = [M, N, O]
physical_therapies = [U, V, W]
prescribed = Function(treatments, bool)
Count([t:treatments], prescribed(t)) == 5
Count([a:antibiotics], prescribed(a)) <= 2
Count([d:dietary_regimens], prescribed(d)) == 1
Implies(Not(prescribed(O)), Not(prescribed(F)))
Implies(prescribed(W), Not(prescribed(F)))
Implies(And(prescribed(N), prescribed(U)), Not(prescribed(G)))
Implies(prescribed(V), And(prescribed(H), prescribed(M)))

solve(Implies(prescribed(O), And(prescribed(U), prescribed(V)))) # (A)
solve(Implies(prescribed(O), And(prescribed(G), prescribed(V)))) # (B)
solve(Implies(prescribed(O), And(prescribed(N), prescribed(U)))) # (C)
solve(Implies(prescribed(O), And(prescribed(U), prescribed(V)))) # (D)
solve(Implies(prescribed(O), And(prescribed(U), prescribed(W)))) # (E)
```

CoT (annotated by our authors)

Nine different treatments are available for a certain illness: three antibiotics—F, G, and H—three dietary regimens—M, N, and O—and three physical therapies—U, V, and W ...... (omitted, see above)
Question: If O is prescribed for a given case, which one of the following is a pair of treatments both of which must also be prescribed for that case?
(A) F, M (B) G, V (C) N, U (D) U, V (E) U, W

```
Let’s first analyze the conditions. We know there must be exactly one dietary regimen
prescribed. If O is a prescribed dietary regimen, then M and N must not be prescribed.
We know V cannot be prescribed unless both H and M are prescribed, so V cannot be
prescribed.
Let’s consider option (A). M must not be prescribed. This option is False.
Let’s consider option (B). V cannot be prescribed. This option is False.
Let’s consider option (C). N must not be prescribed. This option is False.
Let’s consider option (D). V cannot be prescribed. This option is False.
Let’s consider option (E). Let’s assume U is not prescribed. Because exactly 5 of the
treatments will be prescribed, they must be F, G, H, O, and W. In this way both F and
W are prescribed, which contradicts the condition that if W is prescribed, F cannot be
prescribed. So U must be prescribed. We can prescribe G, H, O, U, W. This option is
True.
So the answer is (E).
```

Figure 12: Prompt (excerpt) used for LSAT. Another example can be found in Figure 1. Several of our authors authored different CoT prompts, leading to similar performance (ranging from 20% to 22%).

Prompts for CLUTRR

SatLM

Dorothy took her daughter Michelle and her mother Gabrielle car shopping.
Q: How is [Michelle] related to [Gabrielle]?

```
# [Dorothy] took her daughter [Michelle] and her mother [Gabrielle] car shopping.
relation(Dorothy, Michelle) = (mother, daughter)
relation(Dorothy, Gabrielle) = (daughter, mother)
# How is [Michelle] related to [Gabrielle]?
solve(relation(Michelle, Gabrielle))
```

Teresa and her brother Ellis were having a wonderful time at Disneyland. Ellis asked his grandmother, Molly, to read him a bedtime story. Molly read him Hansel & Gretel, which the boy always loved. Sandra is married to Thomas, the couple welcomed Teresa into the world.
Q: How is [Molly] related to [Sandra]?

```
# [Teresa] and her brother [Ellis] were having a wonderful time at Disneyland.
relation(Teresa, Ellis) = (sister, brother)
# [Ellis] asked his grandmother, [Molly], to read him a bedtime story.
relation(Ellis, Molly) = (grandson, grandmother)
# [Sandra] is married to Thomas, the couple welcomed [Teresa] into the world.
relation(Sandra, Teresa) = (mother, daughter)
# How is [Molly] related to [Sandra]?
solve (relation(Molly, Sandra))
```

ProgLM from [27]

Dorothy took her daughter Michelle and her mother Gabrielle car shopping.
Q: How is [Michelle] related to [Gabrielle]?

```
# To answer this question, we write a program to answer the following subquestions:
# 1. How is [Michelle] related to [Dorothy]? (independent, support: "[Dorothy] took
her daughter [Michelle] and her mother [Gabrielle] car shopping.")
relation(Michelle, Dorothy) = daughter
# 2. How is [Dorothy] related to [Gabrielle]? (independent, support: "[Dorothy] took
her daughter [Michelle] and her mother [Gabrielle] car shopping.")
relation(Dorothy, Gabrielle) = daughter
# 3. Final answer: How is [Michelle] related to [Gabrielle]? (depends on 1, 2)
relation(Michelle, Gabrielle) = relation(Michelle, Dorothy) @ relation(Dorothy,
Gabrielle)
```

Teresa and her brother Ellis were having a wonderful time at Disneyland...... (omitted, see above)
Q: How is [Molly] related to [Sandra]?

```
# To answer this question, we write a program to answer the following subquestions:
# 1. How is [Molly] related to [Ellis]? (independent, support: "[Ellis] asked his
grandmother, [Molly], to read him a bedtime story.")
relation(Molly, Ellis) = grandmother
# 2. How is [Ellis] related to [Teresa]? (independent, support: "[Teresa] and her
brother [Ellis] were having a wonderful time at Disneyland.")
relation(Ellis, Teresa) = brother
# 3. How is [Teresa] related to [Sandra]? (independent, support: "[Sandra] is married
to Thomas, the couple welcomed [Teresa] into the world.")
relation(Teresa, Sandra) = daughter
# 4. Final answer: How is [Molly] related to [Sandra]? (depends on 1, 2, 3)
relation(Molly, Sandra) = relation(Molly, Ellis) @ relation(Ellis, Teresa) @
relation(Teresa, Sandra)
```

Figure 13: Prompt (excerpt) used for CLUTRR.

Prompts for PROOFWRITER

SATLM

Here are some facts and rules:
If someone visits the squirrel and the squirrel visits the rabbit then they are round. All round people are not kind. If someone is round then they chase the rabbit. If someone is red and they chase the rabbit then they visit the dog. If someone is red then they visit the squirrel. If someone visits the squirrel then the squirrel visits the rabbit. the rabbit visits the dog.
the squirrel chases the bald eagle. the squirrel chases the rabbit. the dog sees the bald eagle. the bald eagle does not chase the dog. the bald eagle is red. the squirrel is round. the rabbit does not see the dog. the rabbit sees the bald eagle. the rabbit sees the squirrel. the dog does not see the rabbit. the rabbit does not visit the bald eagle. the dog does not chase the bald eagle.
Q: The statement "The bald eagle visits the dog" is True or False?

```
ForAll([x], Implies(And(visit(x, squirrel), visit(squirrel, rabbit)), round(x)))
ForAll([x], Implies(round(x), Not(kind(x))))
ForAll([x], Implies(round(x), chase(x, rabbit)))
ForAll([x], Implies(And(red(x), chase(x, rabbit)), visit(x, dog)))
ForAll([x], Implies(red(x), visit(x, squirrel)))
ForAll([x], Implies(visit(x, squirrel), visit(squirrel, rabbit)))
chase(squirrel, rabbit)
see(dog, bald_eagle)
Not(chase(bald_eagle, dog))
red(bald_eagle)
round(squirrel)
Not(see(rabbit, dog))
see(rabbit, bald_eagle)
see(rabbit, squirrel)
Not(see(dog, rabbit))
Not(visit(rabbit, bald_eagle))
Not(chase(dog, bald_eagle))

solve(visit(bald_eagle, dog))
```

PROGLM adapted from [8]

Here are some facts and rules:
If someone visits the squirrel and the squirrel visits the rabbit then they are round...... (omitted, see above)
Q: The statement "The bald eagle visits the dog" is True or False?

```
# the bald eagle is red.
bald_eagle_is_red = True
# If someone is red then they visit the squirrel.
bald_eagle_visits_squirrel = bald_eagle_is_red
# If someone visits the squirrel then the squirrel visits the rabbit.
squirrel_visits_rabbit = bald_eagle_visits_squirrel
# If someone visits the squirrel and the squirrel visits the rabbit then they are round.
bald_eagle_is_round = bald_eagle_visits_squirrel and squirrel_visits_rabbit
# If someone is round then they chase the rabbit.
bald_eagle_chases_rabbit = bald_eagle_is_round
# If someone is red and they chase the rabbit then they visit the dog.
bald_eagle_visits_dog = bald_eagle_is_red and bald_eagle_chases_rabbit
# Question: The statement "The bald eagle visits the dog" is True or False?
return bald_eagle_visits_dog
```

Figure 14: Prompt (excerpt) used for PROOFWRITER.

Prompts for COLOREDOBJECT

SATLM

Q: On the table, you see a bunch of objects arranged in a row: a purple paperclip, a pink stress ball, a brown keychain, a green scrunchiephone charger, a mauve fidget spinner, and a burgundy pen. What is the color of the object directly to the right of the stress ball?

```
# What is the color of the object directly to the right of the stress ball?

stress_ball = next(x:objects, name(x) == ’stress ball’)

direct_right = next(x:objects, index(x) - index(stress_ball) == 1)

solve(color(direct_right))
```

PROGLM from [14]

Q: On the table, you see a bunch of objects arranged in a row: a purple paperclip, a pink stress ball, a brown keychain, a green scrunchiephone charger, a mauve fidget spinner, and a burgundy pen. What is the color of the object directly to the right of the stress ball?

```
# Find the index of the stress ball
stress_ball_idx = None
for i, object in enumerate(objects):
  if object[0] == ’stress ball’:
    stress_ball_idx = i
    break


# Find the directly right object
direct_right = objects[i+1]


# Check the directly right object’s color
direct_right_color = direct_right[1]
answer = direct_right_color

return answer
```

Figure 15: Prompt (excerpt) used for COLORED OBJECT.

Prompts for STRUCTUREDREGEX

SATLM

Find the regex for the described patterns. Each regex r can be composed using sub-regexes r1, r2, r3, ...

Pattern:
Three strings separated by semicolons. The first string can either be 579 or 719, the second and third are composed by three digits or three lower case letters that can be followed by a lower case letter, a digit or a capital letter.

```
r = concat(r1,concat(<;>,concat(r2,concat(<;>,r2))))
r1 = or(<579>,<719>)
r2 = concat(or(r3,r4),optional(r5))
r3 = repeat(<num>,3)
r4 = repeat(<low>,3)
r5 = or(<low>,or(<num>,<cap>))
```

PROGLM from [14]

Find the regex for the described patterns.

Pattern:
Three strings separated by semicolons. The first string can either be 579 or 719, the second and third are composed by three digits or three lower case letters that can be followed by a lower case letter, a digit or a capital letter.
Regex:

```
concat(or(<579>,<719>),concat(<;>,concat(concat(or(repeat(<num>,3),repeat(<low>,3)),
optional(or(<low>,or(<num>,<cap>)))),concat(<;>,concat(or(repeat(<num>,3),repeat(<low>,3)),
optional(or(<low>,or(<num>,<cap>))))))))
```

Figure 16: Prompt (excerpt) used for STRUCTUREDREGEX.

Prompts for SAT$_{\text{CoTSolver}}$

SAT$_{\text{CoTSolver}}$ for GSM

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?

```
jason_lollipops_initial = 20
lollipops_given = Variable()
jason_lollipops_after = 12
jason_lollipops_after = jason_lollipops_initial - lollipops_given
result = lollipops_given
solve(result)
# Solve the value of result
# We know jason_lollipops_after = jason_lollipops_initial - lollipops_given
# So lollipops_given = jason_lollipops_initial - jason_lollipops_after = 20 - 12 = 8
# result = lollipops_given = 8
# The answer is 8
```

SAT$_{\text{CoTSolver}}$ for CLUTRR

Dorothy took her daughter Michelle and her mother Gabrielle car shopping.
Q: How is [Michelle] related to [Gabrielle]?

```
# [Dorothy] took her daughter [Michelle] and her mother [Gabrielle] car shopping.
relation(Dorothy, Michelle) = (mother, daughter)
relation(Dorothy, Gabrielle) = (daughter, mother)
# How is [Michelle] related to [Gabrielle]?
solve(relation(Michelle, Gabrielle))
# Solve the problem
# Gabrielle’s daughter is Dorothy, and Dorothy’s daughter is Michelle, so Gabrielle’s
granddaughter is Michelle.
# The answer is granddaughter.
```

Figure 17: Prompt (excerpt) used for SAT$_{\text{CoTSolver}}$.