Fact-Checking Complex Claims with Program-Guided Reasoning

Liangming Pan, Xiaobao Wu, Xinyuan Lu,
Anh Tuan Luu, William Yang Wang, Min-Yen Kan, Preslav Nakov

ACL 2023 (Long Paper)

Presenter: Liangming Pan

What is Fact Checking?

- The proliferation of disinformation in various forms, including propaganda, news, and social media, has made automated fact-checking a crucial application of natural language processing (NLP).

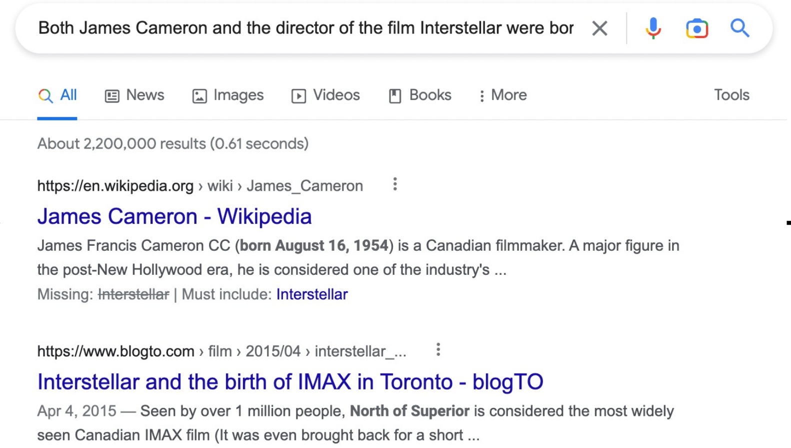
In the language of NLP:

- The goal of fact-checking is, given a claim made by a claimant, to find a collection of evidence and provide a verdict about the claim's veracity based on the evidence. (Glockner et al., 2022)

Verifying Deep Claims

- To verify a real-world claim, we often cannot find a "direct evidence" to support / refute the claim. Instead, it often requires complex, multi-step reasoning.

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.



NO DIRECT EVIDENCE

Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

Where was James Cameron born? → Canada

Who is the director of the film Interstellar? → Christopher Nolan

Where was Christopher Nolan born? → United Kingdom

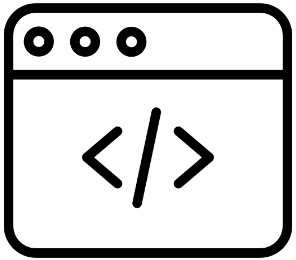
Therefore, the claim is FALSE

Idea: We formulate the above process as Program Execution

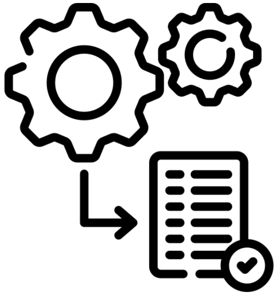
Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

<u>Reasoning Program</u>



<u>Program Execution</u>



LABEL = ✕ **REFUTES**

Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

S1 FACT_1 = **Verify** [James Cameron was born in Canada.]

S2 **ANSWER_1** = **Question** [Who is the director of the film Interstellar?]

S3 FACT_1 = **Verify** [{ANSWER_1} was born in Canada.]

S4 LABEL = **Predict** [{FACT_1} **AND** {FACT_2}]

<u>Reasoning Program</u>

Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

FACT_1 = **Verify** [James Cameron was born in Canada.]

Return Value

Function Call

Function Argument

- **Verify**
- **Question**
- **Predict**

Reasoning Program

Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

S1 FACT_1 = **Verify** [James Cameron was born in Canada.]

S2 **ANSWER_1** = **Question** [Who is the director of the film Interstellar?]

S3 FACT_1 = **Verify** [{ANSWER_1} was born in Canada.]

S4 LABEL = **Predict** [{FACT_1} **AND** {FACT_2}]

Reasoning Program

Functions

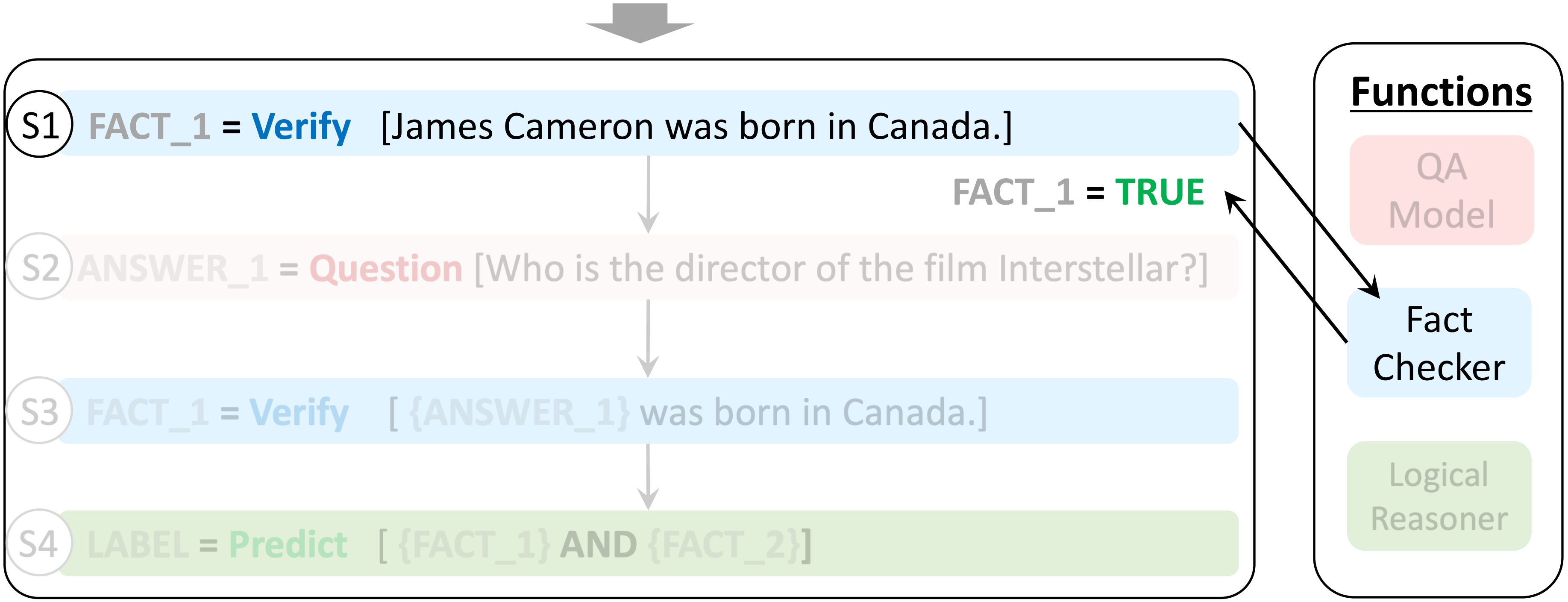
QA Model

Fact Checker

Logical Reasoner

Idea: Program-Guided Reasoning

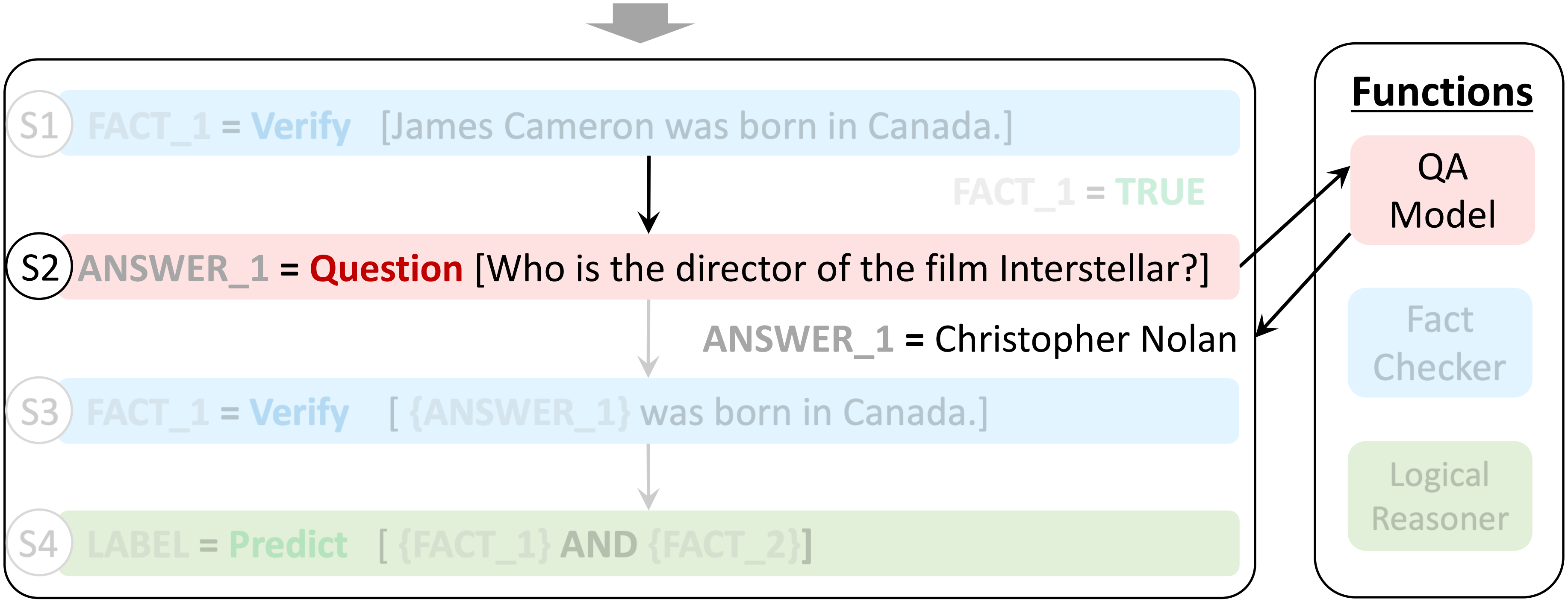
Claim: Both James Cameron and the director of the film Interstellar were born in Canada.



Reasoning Program

Idea: Program-Guided Reasoning

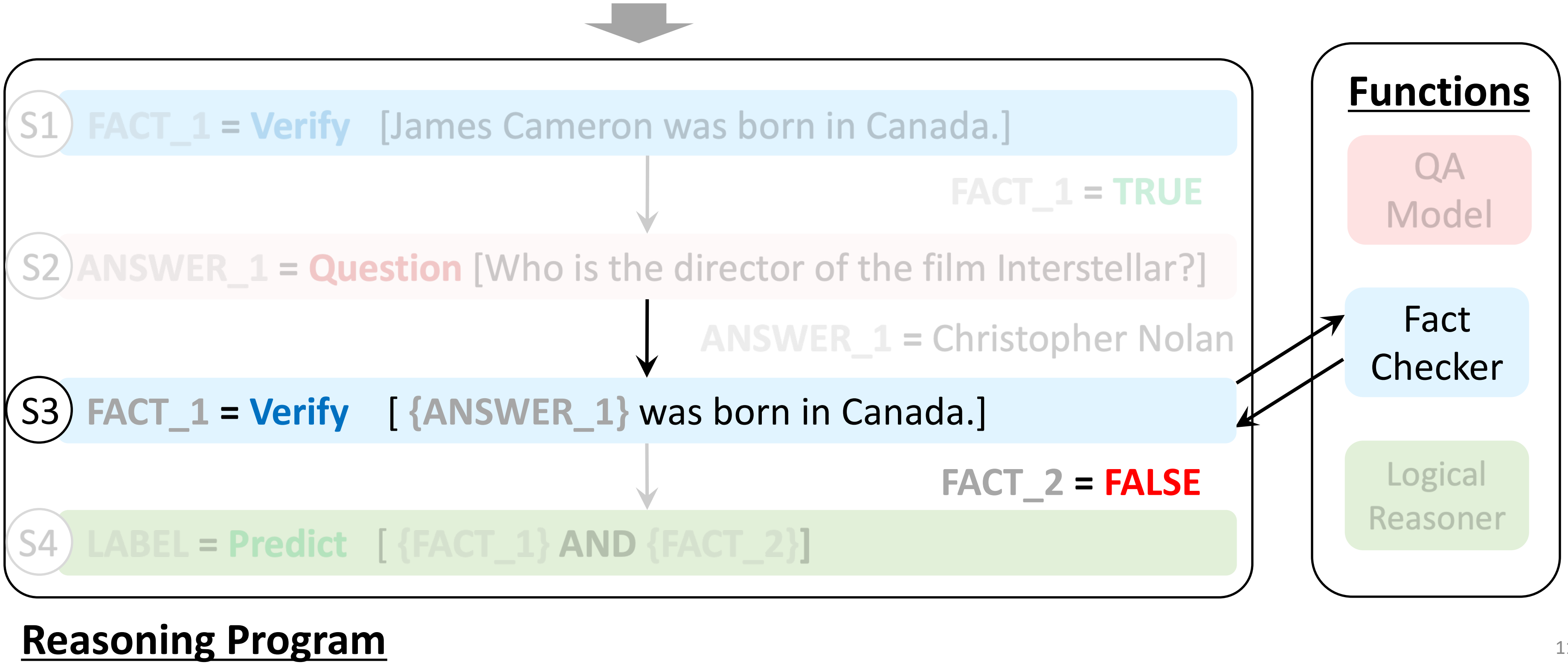
Claim: Both James Cameron and the director of the film Interstellar were born in Canada.



Reasoning Program

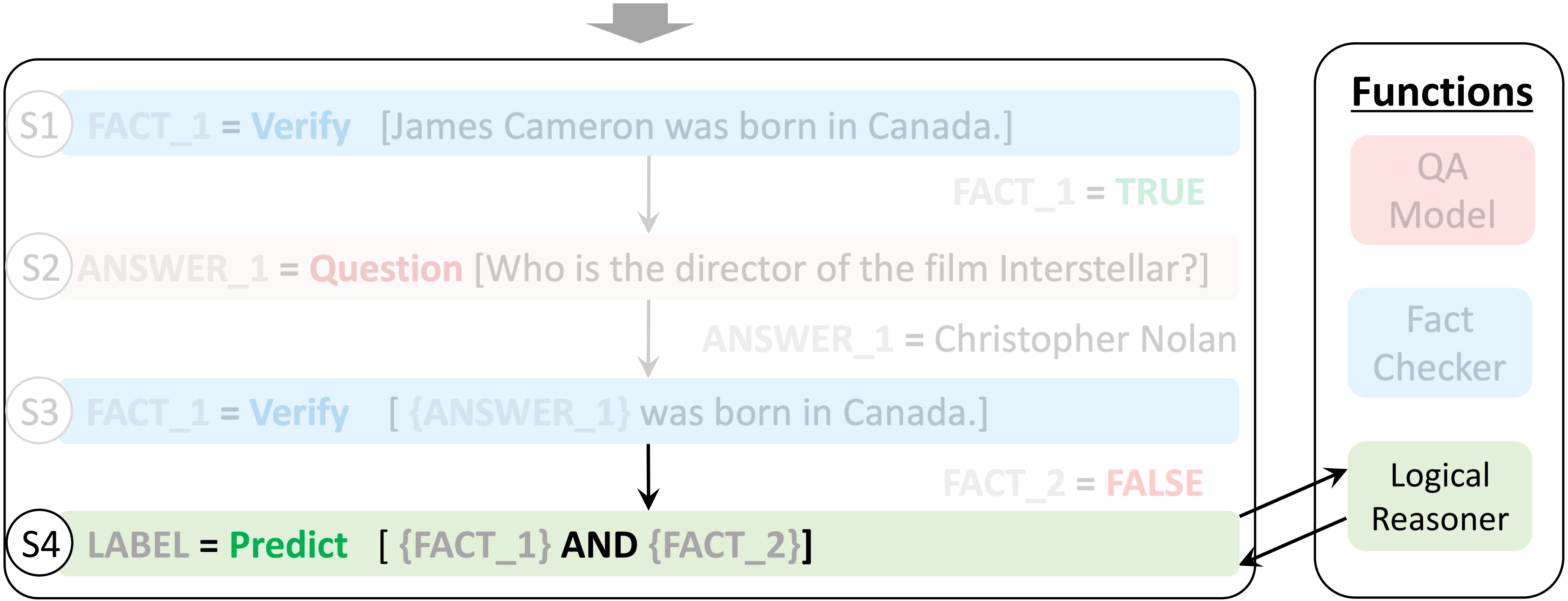
Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.



Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.



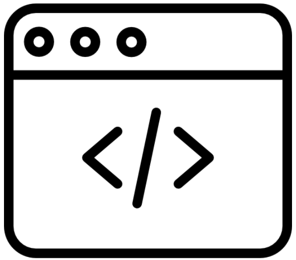
Reasoning Program

LABEL = REFUTES

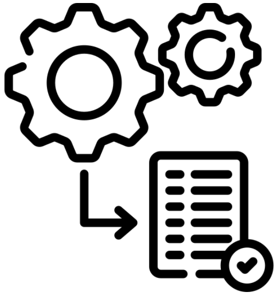
Idea: Program-Guided Reasoning

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.

Reasoning Program



Program Execution

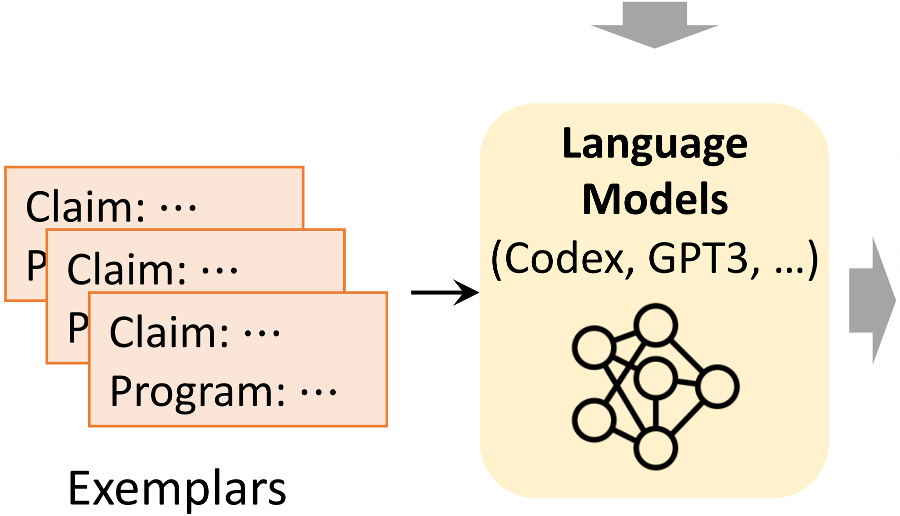


LABEL = ✕ REFUTES

We decouple the program generation and program execution for flexibility and easy debugging.

Reasoning Program Generation

Claim: Both James Cameron and the director of the film Interstellar were born in Canada.




```
'''Generate a python-like program that describes the reasoning steps
   required to verify the claim step-by-step. You can call three functions
   in the program: 1. Question() to answer a question; 2. Verify() to
   verify a simple claim; 3. Predict() to predict the veracity label.'''

# The claim is that Both James Cameron and the director of the film
   Interstellar were born in Canada.
def program():
    fact_1 = Verify("James Cameron was born in Canada.")
    Answer_1 = Question("Who is the director of the film Interstellar?")
    fact_2 = Verify("{Answer_1} was born in Canada.")
    label = Predict(fact_1 and fact_2)

(··· more in-context examples here ···)

# The claim is that <input_claim>
def program():
```

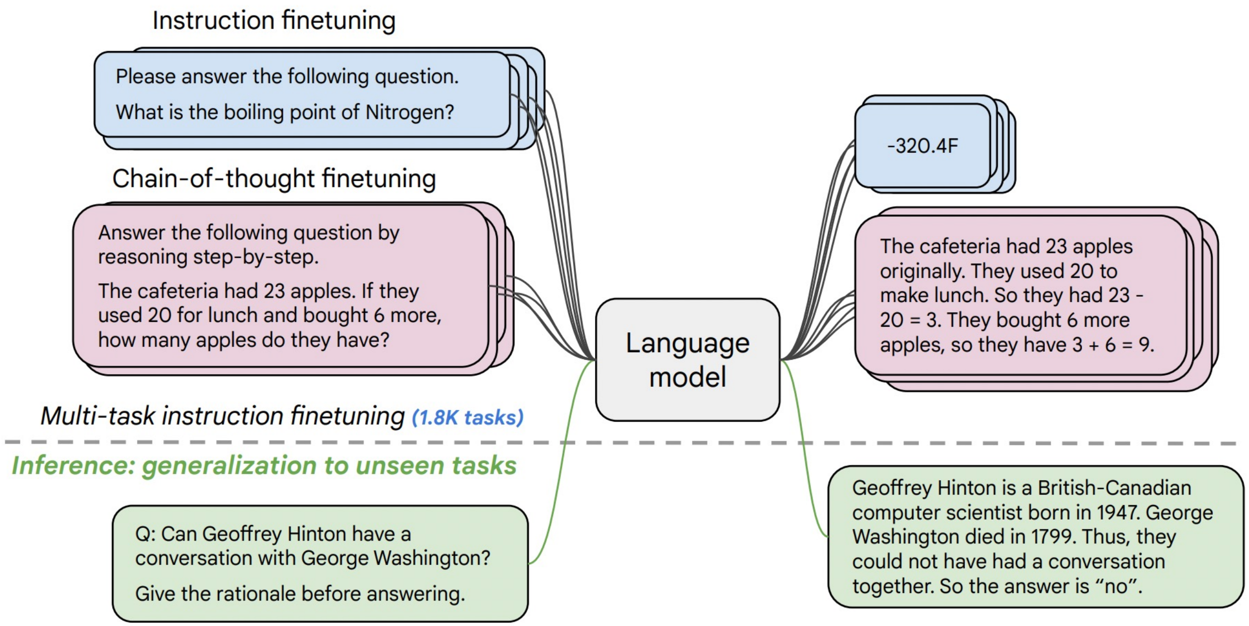
We use in-context learning for data efficiency.

Sub-Functions: QA

Functions

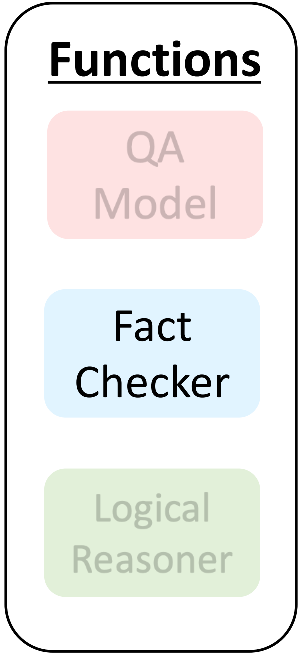
- QA Model
- Fact Checker
- Logical Reasoner

- We base the sub-functions on the FLAN-T5 model, which finetunes T5 with 1.8k finetuning tasks, including chain-of-thought data.

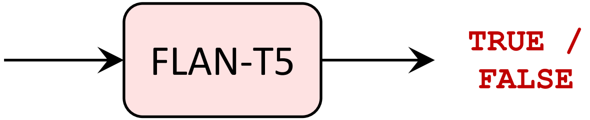


[Scaling Instruction-Finetuned Language Models. (Chung et al., 2022)]

Sub-Functions: QA




```
[<Gold Evidence>,
<Retrieved Evidence>,
<No Evidence>]
Is it true that <Claim>?
True or False?
The answer is:
```



For convenient implementation, we also base the fact-checker with FLAN-T5. The claim is converted into a true/false question.

Evaluation Datasets

- HOVER (Jiang et al., 2020)
  - 1,126 two-hop claims
  - 1,835 three-hop claims
  - 1,039 four-hop claims

Claim: Patrick Carpentier currently drives a Ford Fusion, introduced for model year 2006, in the NASCAR Sprint Cup Series.

Evidence:
Doc A: Ford Fusion is manufactured and marketed by Ford. Introduced for the 2006 model year, …

Doc B: Patrick Carpentier competed in the NASCAR Sprint Cup Series, driving the Ford Fusion. …

Verdict: Supported

- FEVEROUS (Aly et al., 2021)
  - We only selected 2,962 claims that require exclusively textual evidence.

Claim: Red Sundown screenplay was written by Martin Berkeley; based on a story by Lewis B. Patten, who often published under the names Lewis Ford, Lee Leighton and Joseph Wayne.

Evidence:

Page: `wiki/Red_Sundown`
$\mathbf{e}_1$(Introduction):

Red Sundown	
Directed by	Jack Arnold
Produced by	Albert Zugsmith
Screenplay by	Martin Berkeley
Based on	Lewis B. Patten
...	

Page: `wiki/Lewis_B._Patten`
$\mathbf{e}_2$(Introduction): He often published under the names Lewis Ford, Lee Leighton and Joseph Wayne.

Verdict: Supported

Baseline Models

- Pretrained Transformer models
  - BERT-FC (Soleimani et al., 2020)
  - LisT5 (Jiang et al., 2021)

- FC/NLI fine-tuned models
  - RoBERTa-NLI (Nie et al., 2020)
  - DeBERTaV3-NLI (He et al., 2021)
  - MULTIVERS (Wadden et al., 2022)

- In-context learning models
  - FLAN-T5
  - GPT3-Codex

Main Results

Few-shot learning models		HOVER (2-hop)		HOVER (3-hop)		HOVER (4-hop)		FEVEROUS-S	
		Gold	Open	Gold	Open	Gold	Open	Gold	Open
I	BERT-FC (Soleimani et al., 2020)	53.40	50.68	50.90	49.86	50.86	48.57	74.71	51.67
	LisT5 (Jiang et al., 2021)	56.15	52.56	53.76	51.89	51.67	50.46	77.88	54.15
II	RoBERTa-NLI (Nie et al., 2020)	74.62	63.62	62.23	53.99	57.98	52.40	88.28	57.80
	DeBERTaV3-NLI (He et al., 2021)	**77.22**	68.72	65.98	60.76	60.49	56.00	91.98	58.81
	MULTIVERS (Wadden et al., 2022b)	68.86	60.17	59.87	52.55	55.67	51.86	86.03	56.61
III	GPT3-Codex (Chen et al., 2021)	70.63	65.07	66.46	56.63	63.49	57.27	89.77	62.58
	FLAN-T5 (Chung et al., 2022)	73.69	69.02	65.66	60.23	58.08	55.42	90.81	63.73
IV	ProgramFC (N=1)	74.10	69.36	66.13	60.63	65.69	**59.16**	91.77	67.80
	ProgramFC (N=5)	75.65	**70.30**	**68.48**	**63.43**	**66.75**	57.74	**92.69**	**68.06**

PROGRAMFC achieves the best performance on 7 out of 8 evaluations.

Main Results

Few-shot learning models		HOVER (2-hop) Gold	HOVER (2-hop) Open	HOVER (3-hop) Gold	HOVER (3-hop) Open	HOVER (4-hop) Gold	HOVER (4-hop) Open	FEVEROUS-S Gold	FEVEROUS-S Open
I	BERT-FC (Soleimani et al., 2020)	53.40	50.68	50.90	49.86	50.86	48.57	74.71	51.67
	LisT5 (Jiang et al., 2021)	56.15	52.56	53.76	51.89	51.67	50.46	77.88	54.15
II	RoBERTa-NLI (Nie et al., 2020)	74.62	63.62	62.23	53.99	57.98	52.40	88.28	57.80
	DeBERTaV3-NLI (He et al., 2021)	77.22	68.72	65.98	60.76	60.49	56.00	91.98	58.81
	MULTIVERS (Wadden et al., 2022b)	68.86	60.17	59.87	52.55	55.67	51.86	86.03	56.61
III	GPT3-Codex (Chen et al., 2021)	70.63	65.07	66.46	56.63	63.49	57.27	89.77	62.58
	FLAN-T5 (Chung et al., 2022)	73.69	69.02	65.66	60.23	58.08	55.42	90.81	63.73
IV	ProgramFC (N=1)	74.10	69.36	66.13	60.63	65.69	59.16	91.77	67.80
	ProgramFC (N=5)	75.65	70.30	68.48	63.43	66.75	57.74	92.69	68.06

+2.7% +4.3% +14.9%

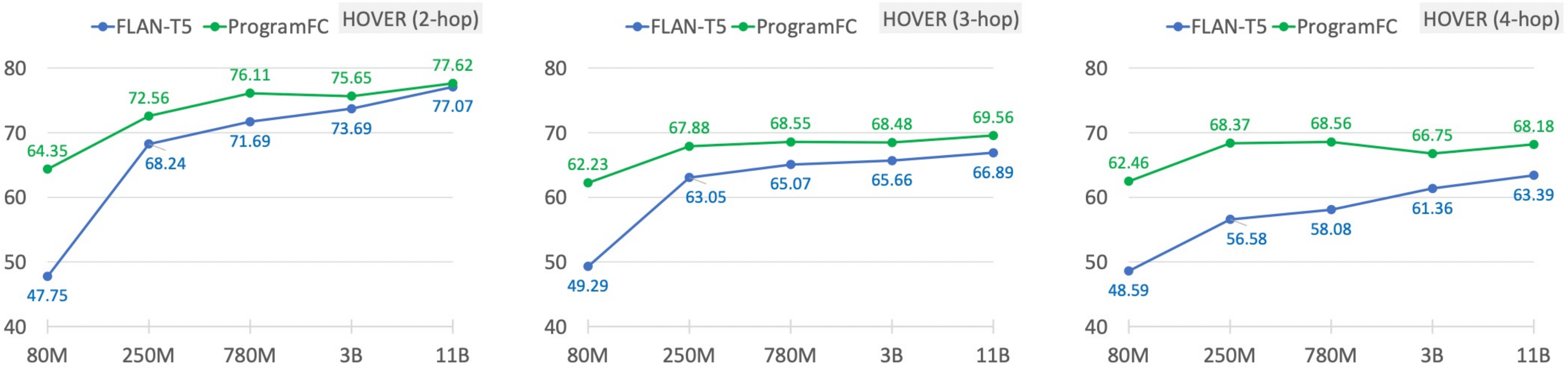
ProgramFC is more effective on deeper claims.

Main Results

Few-shot learning models		HOVER (2-hop)		HOVER (3-hop)		HOVER (4-hop)		FEVEROUS-S	
		Gold	Open	Gold	Open	Gold	Open	Gold	Open
I	BERT-FC (Soleimani et al., 2020)	53.40	50.68	50.90	49.86	50.86	48.57	74.71	51.67
	ListT5 (Jiang et al., 2021)	56.15	52.56	53.76	51.89	51.67	50.46	77.88	54.15
II	RoBERTa-NLI (Nie et al., 2020)	74.62	63.62	62.23	53.99	57.98	52.40	88.28	57.80
	DeBERTaV3-NLI (He et al., 2021)	**77.22**	68.72	65.98	60.76	60.49	56.00	91.98	58.81
	MULTIVERS (Wadden et al., 2022b)	68.86	60.17	59.87	52.55	55.67	51.86	86.03	56.61
III	GPT3-Codex (Chen et al., 2021)	70.63	65.07	66.46	56.63	63.49	57.27	89.77	62.58
	FLAN-T5 (Chung et al., 2022)	73.69	69.02	65.66	60.23	58.08	55.42	90.81	63.73
IV	ProgramFC (N=1)	74.10	69.36	66.13	60.63	65.69	**59.16**	91.77	67.80
	ProgramFC (N=5)	75.65	**70.30**	**68.48**	**63.43**	**66.75**	57.74	**92.69**	**68.06**

Aggregating reasoning programs is helpful.

How Reasoning Program Helps?



- PROGRAMFC v.s. FLAN-T5 using different language model sizes.
- The performance of the end-to-end FLAN-T5 model decreases significantly with decreasing model size. However, this trend is less obvious for PROGRAMFC.

The high-level planning offered by reasoning programs alleviates the demand on strong, large-scale models.

How Reasoning Program Helps?

- In the open-book setting, ProgramFC significantly outperforms one-step retrieval.

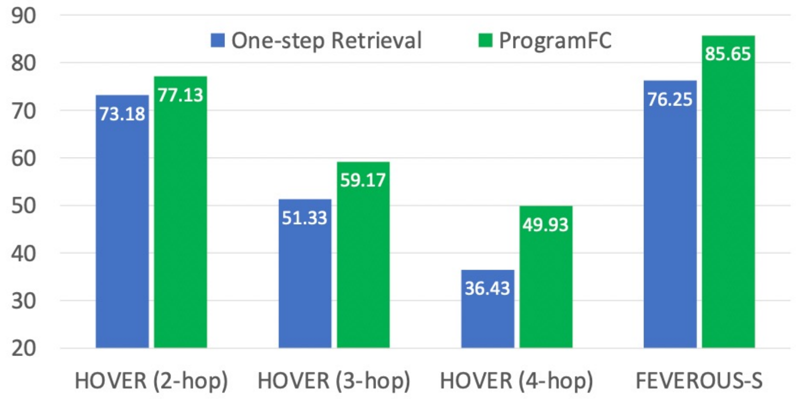


Figure 5: Retrieval recall@10 for the one-step retrieval and the iterative retrieval in PROGRAMFC.

Intuition: Some information may not be present in the original claim, but is only revealed during the reasoning process (e.g., "Christopher Nolan" in the previous example).

Iteratively retrieving information guided by the reasoning program leads to better results.

Reasoning Program Evaluation

- Correct Examples

```
Claim:
Tritonia and Phyteuma are both names for a plant genus.

Predicted Program:
fact_1 = Verify("Tritonia is a name for a plant genus.")
fact_2 = Verify("Phyteuma is a name for a plant genus.")
label = Predict(fact_1 and fact_2)
```

```
Claim:
The country that Fujairah College is located in had a 2013 population of 9.2 million until it was
hit by the plague in 1483 when the population was halved.

Predicted Program:
answer_1 = Question("Which country is Fujairah College located in?")
fact_1 = Verify("{answer_1} had a 2013 population of 9.2 million.")
fact_2 = Verify("{answer_1} was hit by the plague in 1483.")
fact_3 = Verify("The population of {answer_1} was halved in 1483.")
label = Predict(fact_1 and fact_2 and fact_3)
```

Reasoning Program Evaluation

- Wrong Examples

Semantic Error — Token: incorrect or missing arguments/variables

Example 1:

Bitter Jester and The Future of Food are not both documentaries about food.

Predicted Program:

```
fact_1 = Verify("Bitter Jester is a documentary about food.")
fact_2 = Verify("The Future of Food is a documentary about food.")
label = Predict(fact_1 and fact_2) → label = Predict(not (fact_1 and fact_2))
```

Semantic Error — Structure: incorrect program structure

Example 3:

Richard Strauss, German composer of Die Nacht, composed another well-established opera called Der Rosenkavalier.

Predicted Program:

```
fact_1 = Verify("Richard Strauss, German composer of Die Nacht, composed another well-established
                 opera called Der Rosenkavalier.")
label = Predict(fact_1)

→

fact_1 = Verify("Richard Strauss is a German composer of Die Nacht.")
fact_2 = Verify("Richard Strauss composed a well-established opera called Der Rosenkavalier.")
label = Predict(fact_1 and fact_2)
```

Reasoning Program Evaluation

- Wrong Examples

Semantic Error — Subtask: missing / redundant / incorrect sub-task calls

Example 5:
The musician, who founded Morningwood with Max Green, is older than Max Green.

Predicted Program:

```
answer_1 = Question("Who founded Morningwood with Max Green?")
answer_2 = Question("When was Max Green born?")
~~answer_3 = Question("When was the musician born?")~~
fact_1 = Verify("{answer_3} is older than {answer_2}.") → {answer_1} is older than {answer_2}.
label = Verify(fact_1)
```

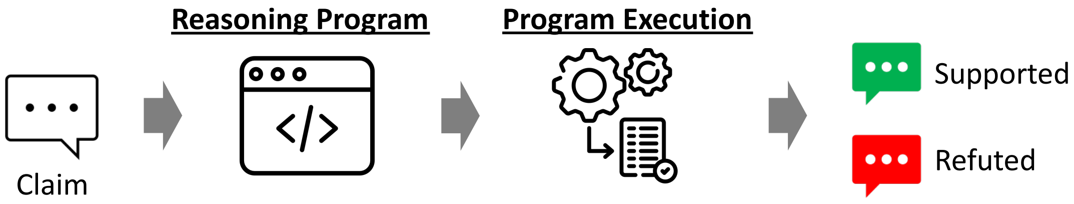
Error Type	Proportion (%)		
	2-hop	3-hop	4-hop
Syntax error	**0%**	**0%**	**0%**
Semantic error	**29%**	**38%**	**77%**
Token	8%	20%	18%
Structure	19%	13%	57%
Subtask	2%	5%	2%
Incorrect execution	**71%**	**62%**	**23%**

Summary

We talked about how to build a fact-checking system that are:

- **Data Efficiency**
  - Build a model with minimal or no training data.
- **Explanablility**
  - Provide a clear explanation of its reasoning process.
- **Deep Reasoning**
  - Collect multiple pieces of evidence and applying complex reasoning.

Our solution: **Program-guided Reasoning.**



Homepage

Thanks!

Github

Any questions?

Liangming Pan
Email: liangmingpan@ucsb.edu