CoqPilot

a plugin for LLM-based generation of proofs

Andrei Kozyrev

Gleb Solovev

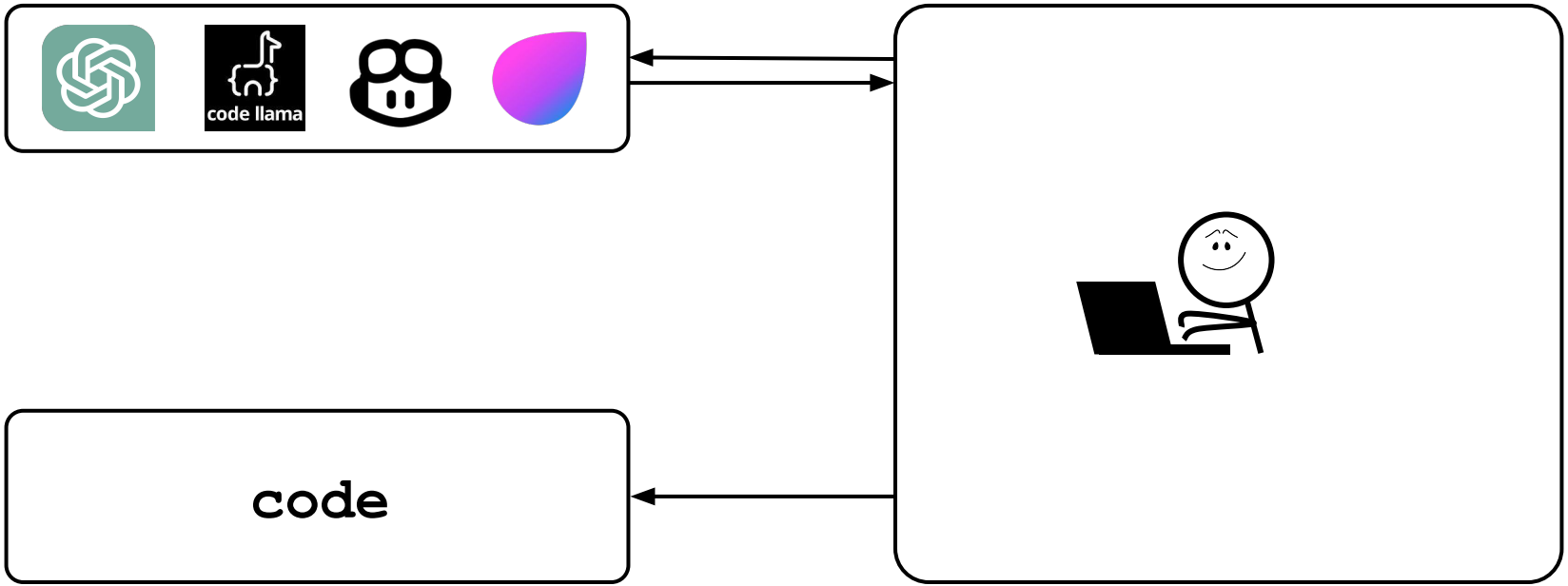
Nikita Khramov

Anton Podkopaev

Programming Languages and Program Analysis Lab (PLAN), JetBrains Research

September, 2024

LLMs are used more and more for code generation



Do Users Write More Insecure Code with AI Assistants?

Neil Perry *
Stanford University

Megha Srivastava *
Stanford University

Deepak Kumar
Stanford University

Dan Boneh
Stanford University

Do Users Write More Insecure Code with AI Assistants?

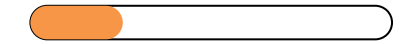
Neil Perry *
Stanford University

Megha Srivastava *
Stanford University

Deepak Kumar
Stanford University

Dan Boneh
Stanford University

introduce security ***vulnerabilities***



Do Users Write More Insecure Code with AI Assistants?

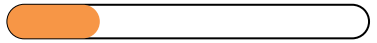
Neil Perry *
Stanford University

Megha Srivastava *
Stanford University

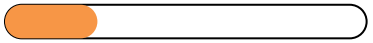
Deepak Kumar
Stanford University

Dan Boneh
Stanford University

introduce security ***vulnerabilities***



rate their ***insecure*** code ***as secure***



Generate a list sorting function in Python without using any library functions

```python
def custom_sort(arr):
    n = len(arr)
    for i in range(n):
        # Last i elements are already sorted
        for j in range(0, n - i - 1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
    return arr
```

Generate a list sorting function in Python without using any library functions


```
def custom_sort(arr):
    # Last i elements are already sorted
    for j in range(0, n - i - 1):
      # Traverse the array from 0 to n-i-1
      # Swap if the element found is greater
      # than the next element
      if arr[j] > arr[j + 1]:
        arr[j], arr[j + 1] = arr[j + 1], arr[j]
  return arr
```

Not only **familiar** code is needed

Generate a list sorting function in Python without using any library functions


```
def custom_sort(arr):
    # Last i elements are already sorted
    for j in range(0, n - i - 1):
      # Traverse the array from 0 to n-i-1
      # Swap if the element found is greater
      # than the next element
      if arr[j] > arr[j + 1]:
        arr[j], arr[j + 1] = arr[j + 1], arr[j]
  return arr
```

Not only **familiar** code is needed

Testing may **not** be **sufficient** (eg, concurrency)

Generate a list sorting function in Python
without using any library functions

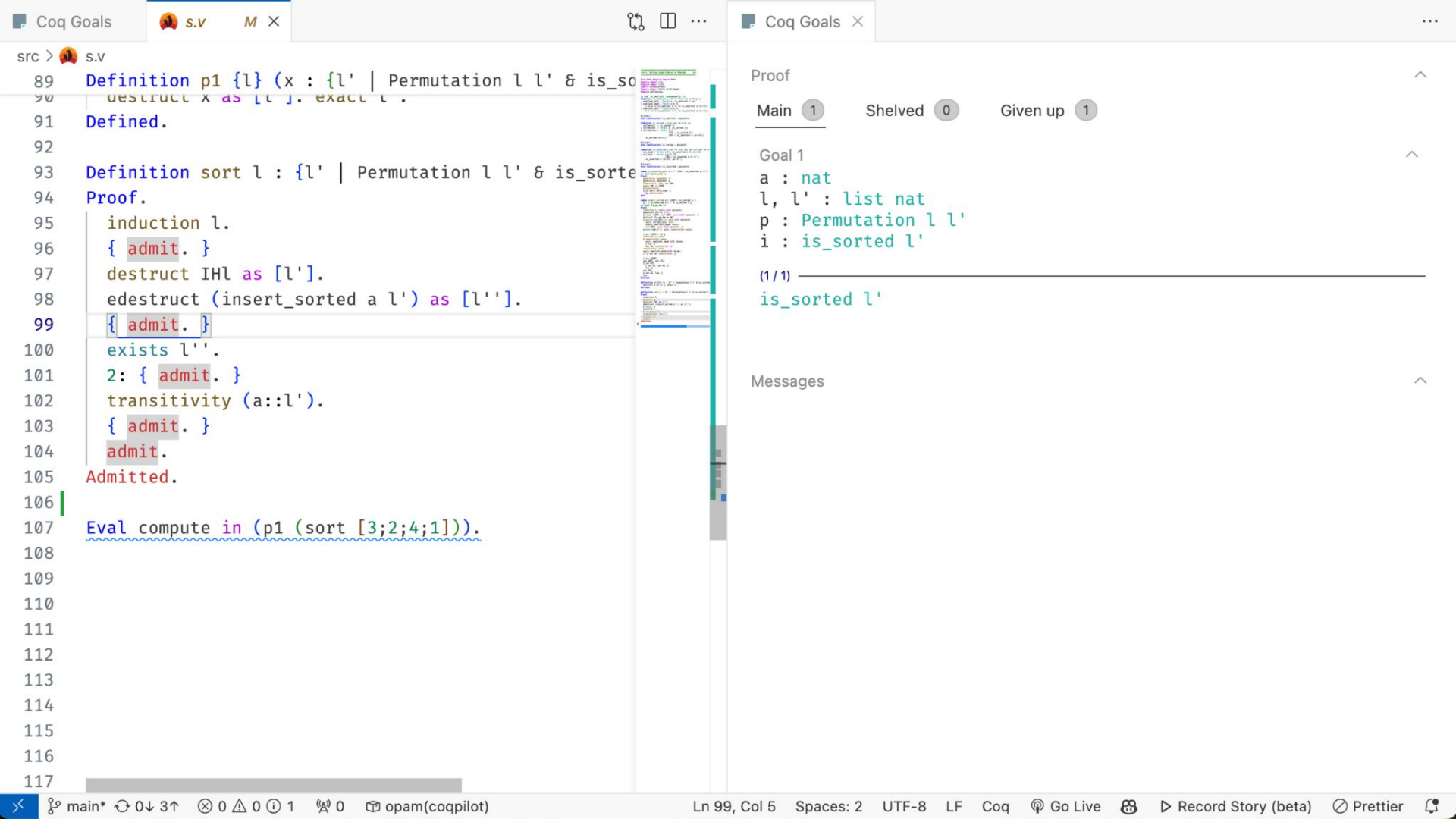

```
def custom_sort(arr):
    # Traverse the array from 0 to n-i-1
    # Swap if the element found is greater
    # than the next element
    if arr[j] > arr[j + 1]:
      arr[j], arr[j + 1] = arr[j + 1], arr[j]
  return arr
```

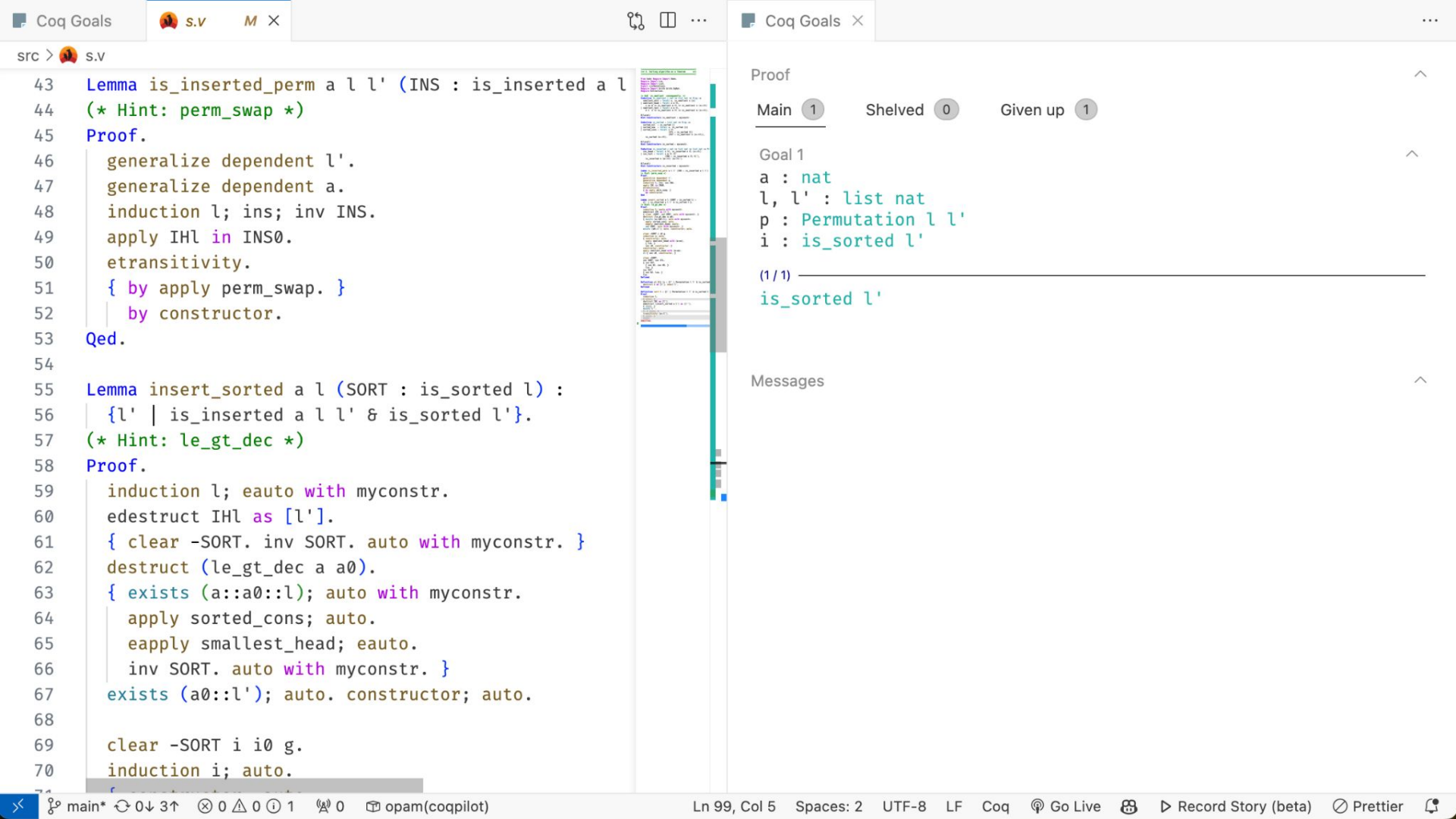
Not only **familiar** code is needed

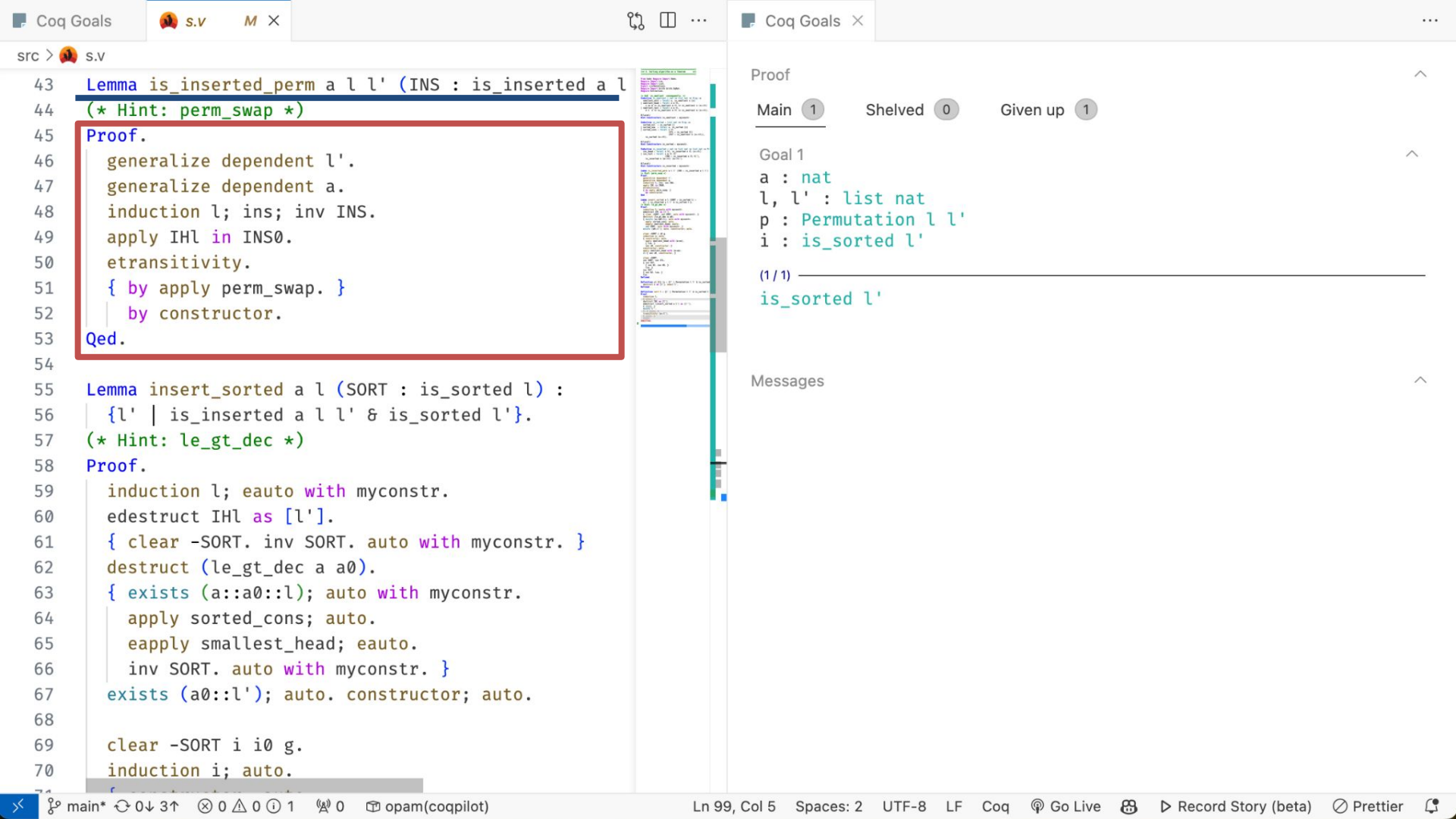
Testing may **not** be **sufficient** (eg, concurrency)

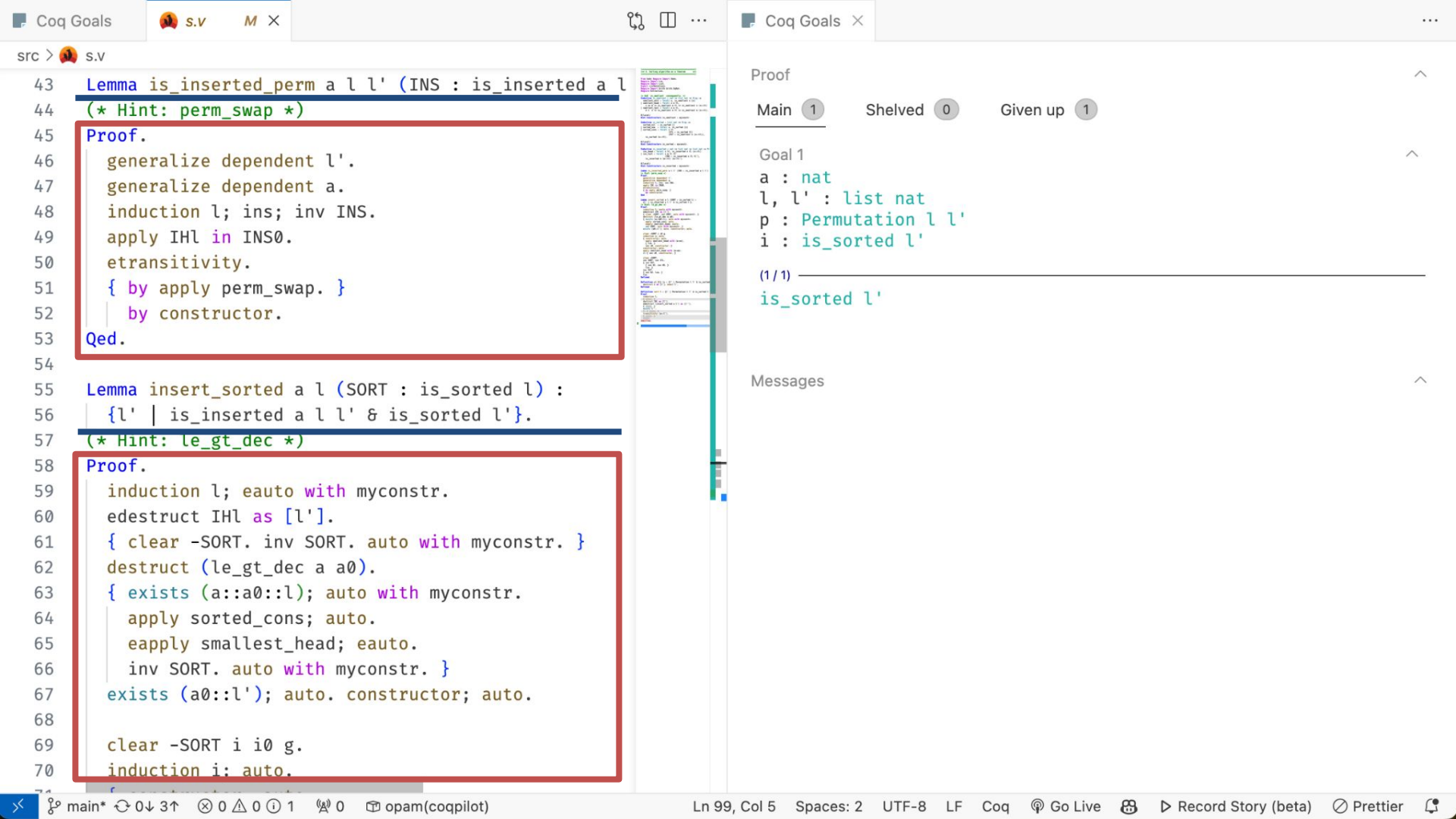
Plain **English** is **hard** to debug and **imprecise**

CoqPilot


Coq Goals
s.v
M
src
s.v
Definition p1 {l} (x : {l' | Permutation l l' & is_so
destruct x as [l']. exact l'.
Defined.
Definition sort l : {l' | Permutation l l' & is_sorte
Proof.
induction l.
{ admit. }
destruct IHl as [l'].
edestruct (insert_sorted a l') as [l''].
{ admit. }
exists l''.
2: { admit. }
transitivity (a::l').
{ admit. }
admit.
Admitted.
Eval compute in (p1 (sort [3;2;4;1])).
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
0↓ 3↑
0 0 1
0
opam(coqpilot)
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier


Coq Goals
s.v
M
src
s.v
Lemma is_inserted_perm a l l' (INS : is_inserted a l
(* Hint: perm_swap *)
Proof.
generalize dependent l'.
generalize dependent a.
induction l; ins; inv INS.
apply IHl in INS0.
etransitivity.
{ by apply perm_swap. }
by constructor.
Qed.
Lemma insert_sorted a l (SORT : is_sorted l) :
{l' | is_inserted a l l' & is_sorted l'}.
(* Hint: le_gt_dec *)
Proof.
induction l; eauto with myconstr.
edestruct IHl as [l'].
{ clear -SORT. inv SORT. auto with myconstr. }
destruct (le_gt_dec a a0).
{ exists (a::a0::l); auto with myconstr.
apply sorted_cons; auto.
eapply smallest_head; eauto.
inv SORT. auto with myconstr. }
exists (a0::l'); auto. constructor; auto.
clear -SORT i i0 g.
induction i; auto.
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
0↓ 3↑
0 0 1
0
opam(coqpilot)
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier


Coq Goals
s.v
M
src
s.v
Lemma is_inserted_perm a l l' (INS : is_inserted a l
(* Hint: perm_swap *)
Proof.
generalize dependent l'.
generalize dependent a.
induction l; ins; inv INS.
apply IHl in INS0.
etransitivity.
{ by apply perm_swap. }
by constructor.
Qed.
Lemma insert_sorted a l (SORT : is_sorted l) :
{l' | is_inserted a l l' & is_sorted l'}.
(* Hint: le_gt_dec *)
Proof.
induction l; eauto with myconstr.
edestruct IHl as [l'].
{ clear -SORT. inv SORT. auto with myconstr. }
destruct (le_gt_dec a a0).
{ exists (a::a0::l); auto with myconstr.
apply sorted_cons; auto.
eapply smallest_head; eauto.
inv SORT. auto with myconstr. }
exists (a0::l'); auto. constructor; auto.
clear -SORT i i0 g.
induction i; auto.
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
0↓ 3↑
0 0 1
0
opam(coqpilot)
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier


Coq Goals
s.v
M
src
s.v
Lemma is_inserted_perm a l l' (INS : is_inserted a l
(* Hint: perm_swap *)
Proof.
generalize dependent l'.
generalize dependent a.
induction l; ins; inv INS.
apply IHl in INS0.
etransitivity.
{ by apply perm_swap. }
by constructor.
Qed.
Lemma insert_sorted a l (SORT : is_sorted l) :
{l' | is_inserted a l l' & is_sorted l'}.
(* Hint: le_gt_dec *)
Proof.
induction l; eauto with myconstr.
edestruct IHl as [l'].
{ clear -SORT. inv SORT. auto with myconstr. }
destruct (le_gt_dec a a0).
{ exists (a::a0::l); auto with myconstr.
apply sorted_cons; auto.
eapply smallest_head; eauto.
inv SORT. auto with myconstr. }
exists (a0::l'); auto. constructor; auto.
clear -SORT i i0 g.
induction i; auto.
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
0↓ 3↑
0 0 1
0
opam(coqpilot)
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier

s.v M

src > s.v

```
Definition sort l : {l' | Permutation l l' & is_sorted l'}.
Proof.
  induction l.
  { admit. }
  destruct IHl as [l'].
  edestruct (insert_sorted a l') as [l''].
  { admit. }
  exists l''.
  2: { admit. }
  transitivity (a::l').
  { admit. }
  admit.
Admitted.

Eval compute in (p1 (sort [3;2;4;1])).
```

Coq Goals

Proof

Not in proof mode

Messages

```
= let (x, _, _) := sort [3; 2; 4; 1] in x
: list nat
```

main* 0↓ 3↑ Launchpad ⊗ 0 ⚠ 0 ⓘ 1 0 opam(coqpilot) vscoq-language-server 2.1.0, coq 8.19.0 Go Live Record Story (beta)

s.v M

src > s.v

```
Definition sort l : {l' | Permutation l l' & is_sorted l'}.
Proof.
  induction l.
  { admit. }
  destruct IHl as [l'].
  edestruct (insert_sorted a l') as [l''].
  { admit. }
  exists l''.
  2: { admit. }
  transitivity (a::l').
  { admit. }
  admit.
Admitted.

Eval compute in (p1 (sort [3;2;4;1])).
```

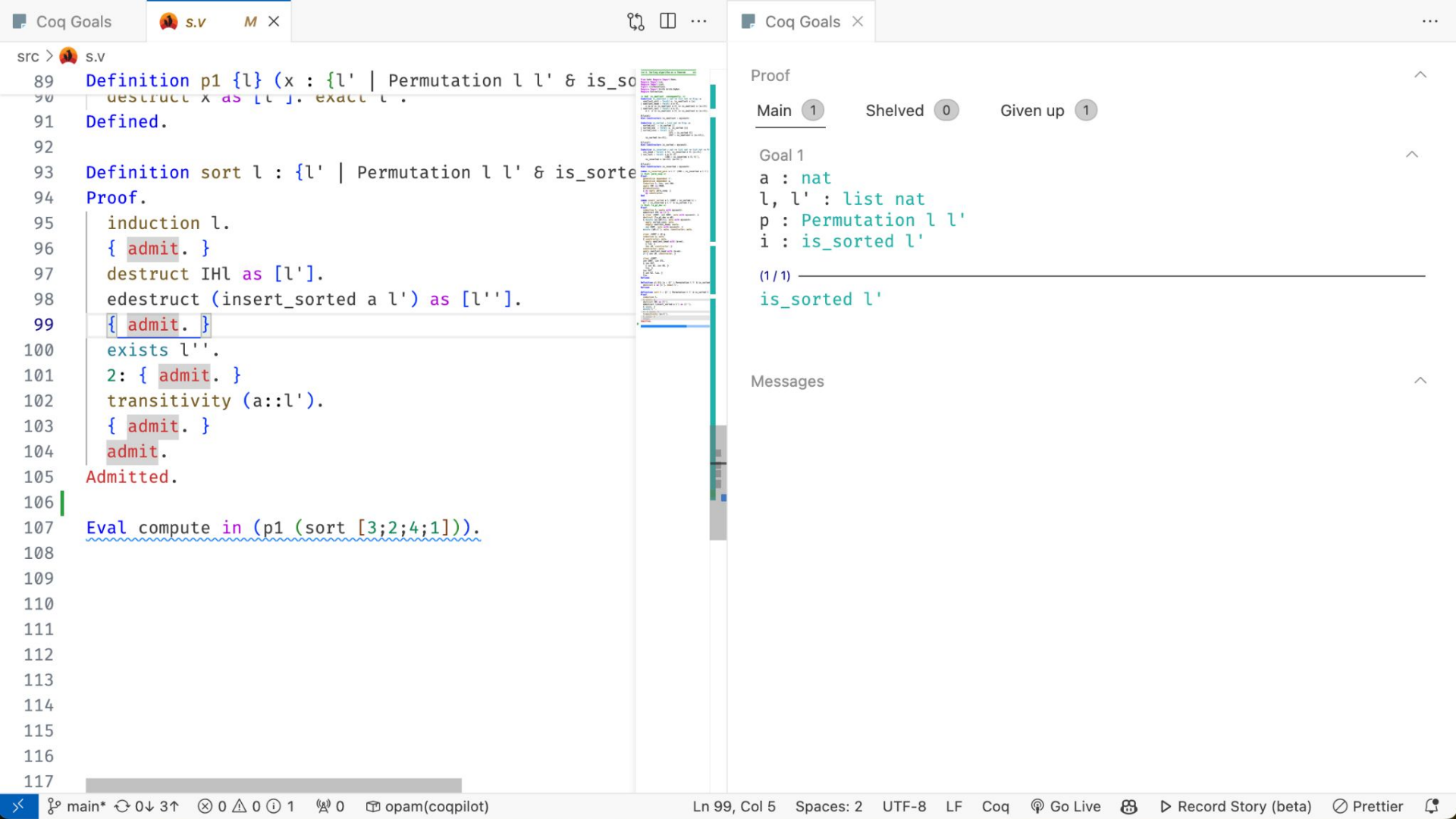
Coq Goals

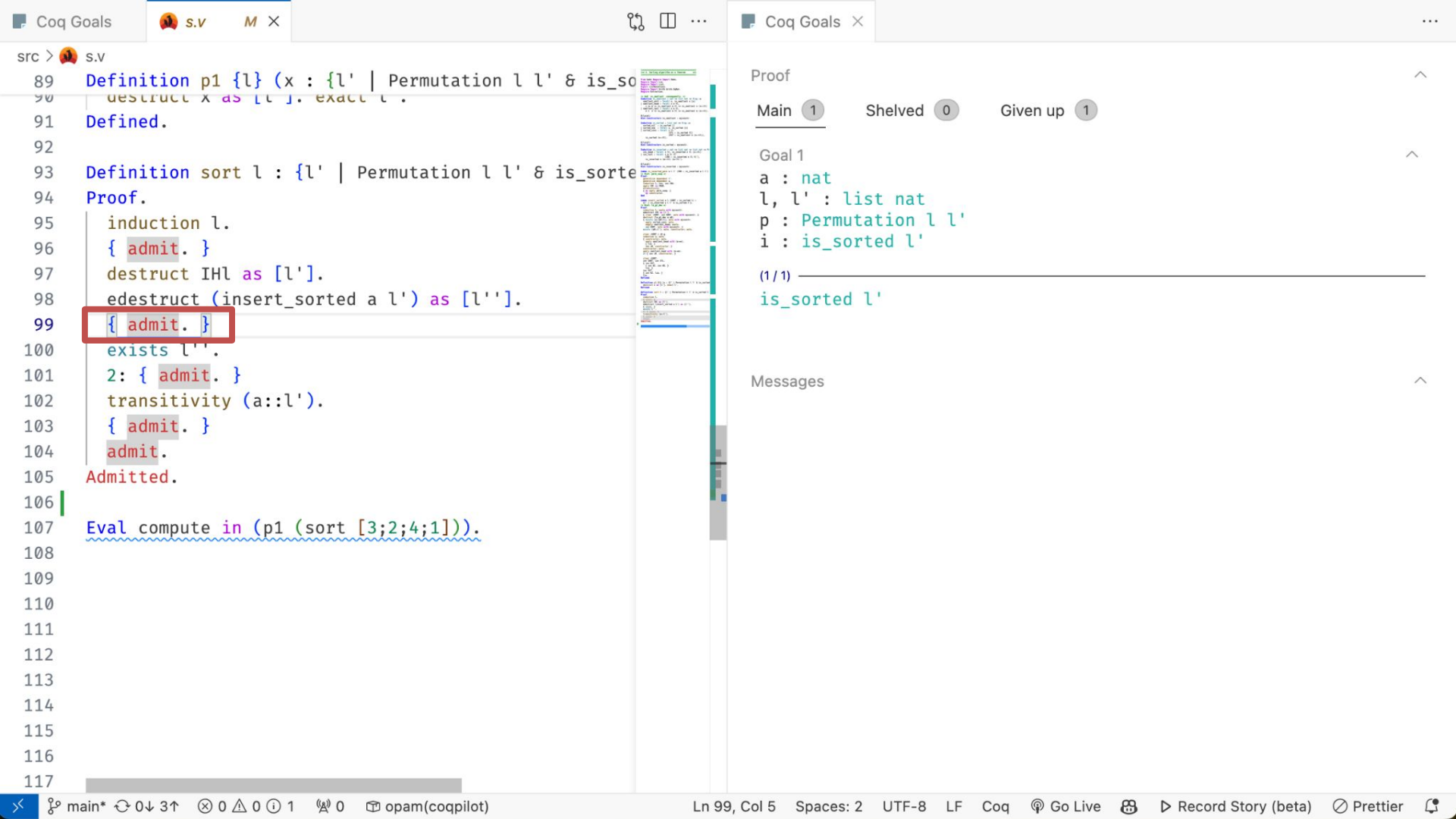
Proof

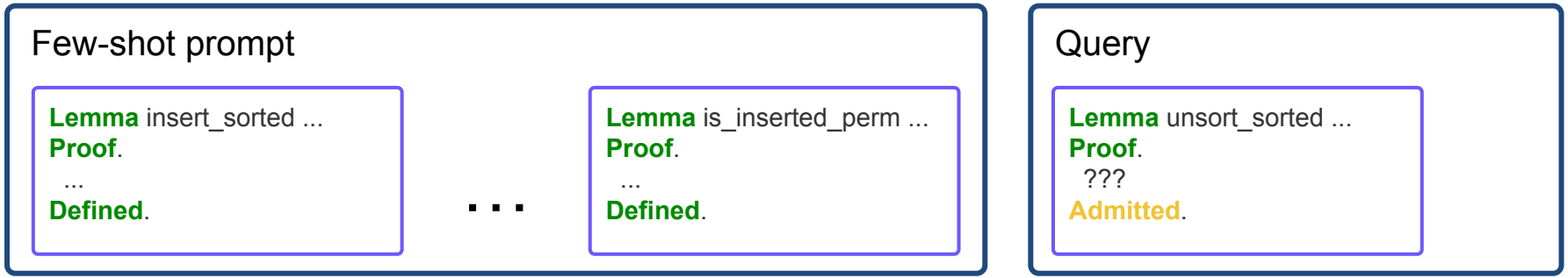
Not in proof mode

Messages

```
= let (x, _, _) := sort [3; 2; 4; 1] in x
: list nat
```


Coq Goals
s.v
M
src
s.v
Definition p1 {l} (x : {l' | Permutation l l' & is_so
destruct x as [l']. exact l'.
Defined.
Definition sort l : {l' | Permutation l l' & is_sorte
Proof.
induction l.
{ admit. }
destruct IHl as [l'].
edestruct (insert_sorted a l') as [l''].
{ admit. }
exists l''.
2: { admit. }
transitivity (a::l').
{ admit. }
admit.
Admitted.
Eval compute in (p1 (sort [3;2;4;1])).
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
0↓ 3↑
0 0 1
0
opam(coqpilot)
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier


Coq Goals
s.v
M
src > s.v
89 Definition p1 {l} (x : {l' | Permutation l l' & is_so
91 Defined.
93 Definition sort l : {l' | Permutation l l' & is_sorte
94 Proof.
95 induction l.
96 { admit. }
97 destruct IHl as [l'].
98 edestruct (insert_sorted a l') as [l''].
99 { admit. }
100 exists l''.
101 2: { admit. }
102 transitivity (a::l').
103 { admit. }
104 admit.
105 Admitted.
107 Eval compute in (p1 (sort [3;2;4;1])).
Coq Goals
Proof
Main 1
Shelved 0
Given up 1
Goal 1
a : nat
l, l' : list nat
p : Permutation l l'
i : is_sorted l'
(1 / 1)
is_sorted l'
Messages
main*
Ln 99, Col 5
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier


Few-shot prompt
Lemma insert_sorted ...
Proof.
...
Defined.
. . .
Lemma is_inserted_perm ...
Proof.
...
Defined.
Query
Lemma unsort_sorted ...
Proof.
???
Admitted.

Few-shot prompt

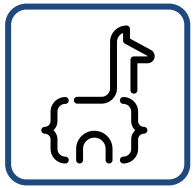
Lemma insert_sorted ...
Proof.
...
Defined.

...

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

Lemma unsort_sorted ...
Proof.
???
Admitted.



...


Few-shot prompt

Lemma insert_sorted ...
Proof.
...
Defined.

. . .

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

Lemma unsort_sorted ...
Proof.
???
Admitted.



. . .


Few-shot prompt

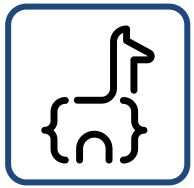
Lemma insert_sorted ...
Proof.
...
Defined.

...

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

Lemma unsort_sorted ...
Proof.
???
Admitted.



...


Few-shot prompt

Lemma insert_sorted ...
Proof.
...
Defined.

...

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

Lemma unsort_sorted ...
Proof.
???
Admitted.



...

Checker

Lemma unsort_sorted
Proof 1.
…
Defined.

Lemma unsort_sorted
Proof 2.
…
Defined.

Few-shot prompt

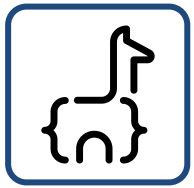
Lemma insert_sorted ...
Proof.
...
Defined.

...

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

Lemma unsort_sorted ...
Proof.
???
Admitted.



...


Checker

Lemma unsort_sorted
Proof 1.
…
Defined.

Lemma unsort_sorted
Proof 2.
…
Defined.

Few-shot prompt

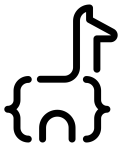
Lemma insert_sorted ...
Proof.
...
Defined.

...

Lemma is_inserted_perm ...
Proof.
...
Defined.

Query

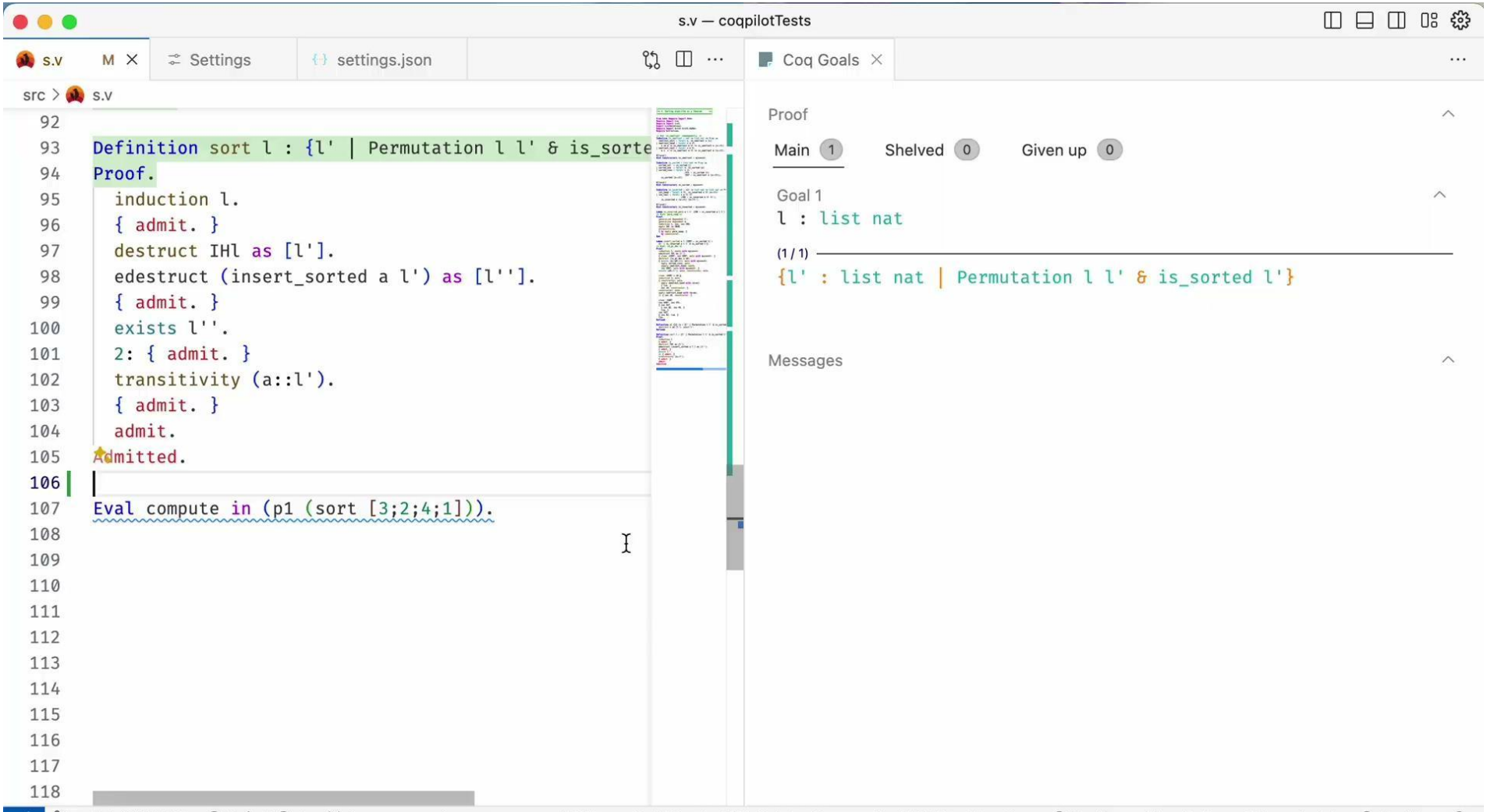
Lemma unsort_sorted ...
Proof.
???
Admitted.



...


Checker

Lemma unsort_sorted
Proof 1.
…
Defined.

Lemma unsort_sorted
Proof 2.
…
Defined.


s.v — coqpilotTests
s.v M
Settings
settings.json
Coq Goals
src > s.v
92
93 Definition sort l : {l' | Permutation l l' & is_sorte
94 Proof.
95 induction l.
96 { admit. }
97 destruct IHl as [l'].
98 edestruct (insert_sorted a l') as [l''].
99 { admit. }
100 exists l''.
101 2: { admit. }
102 transitivity (a::l').
103 { admit. }
104 admit.
105 Admitted.
106
107 Eval compute in (p1 (sort [3;2;4;1])).
108
109
110
111
112
113
114
115
116
117
118
Proof
Main 1
Shelved 0
Given up 0
Goal 1
l : list nat
(1 / 1)
{l' : list nat | Permutation l l' & is_sorted l'}
Messages
main* 0↓ 3↑ 0 0 1 0
vscoq-language-server 2.1.0, coq 8.19.0
Ln 106, Col 1
Spaces: 2
UTF-8
LF
Coq
Go Live
Record Story (beta)
Prettier

Proof repair

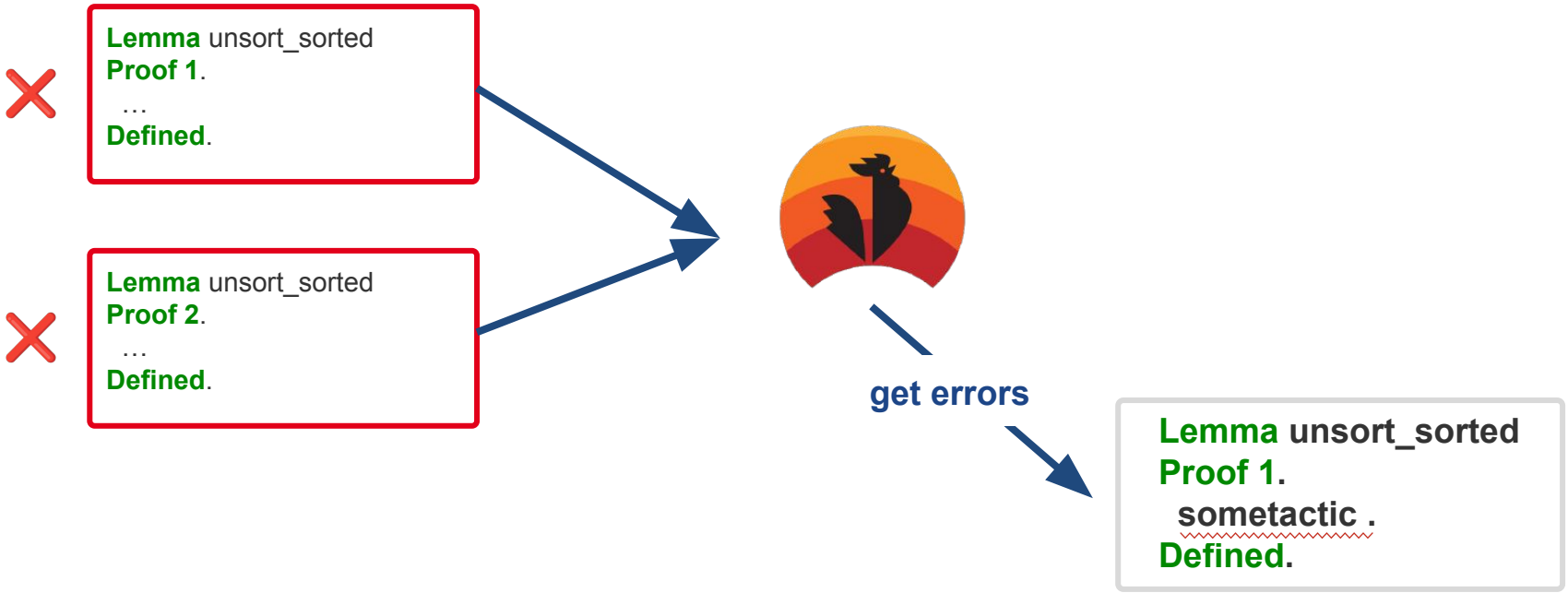
❌

```
Lemma unsort_sorted
Proof 1.
  …
Defined.
```

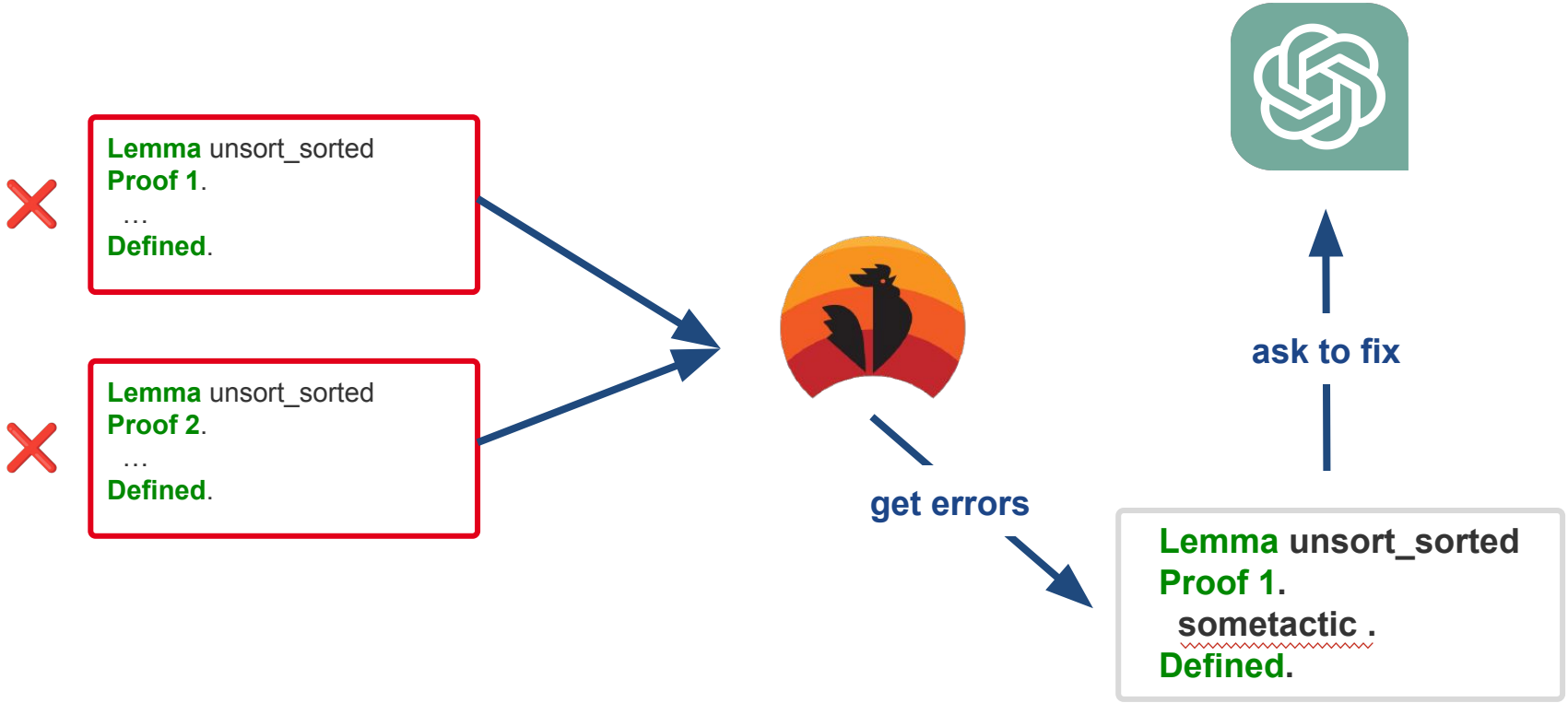
❌

```
Lemma unsort_sorted
Proof 2.
  …
Defined.
```

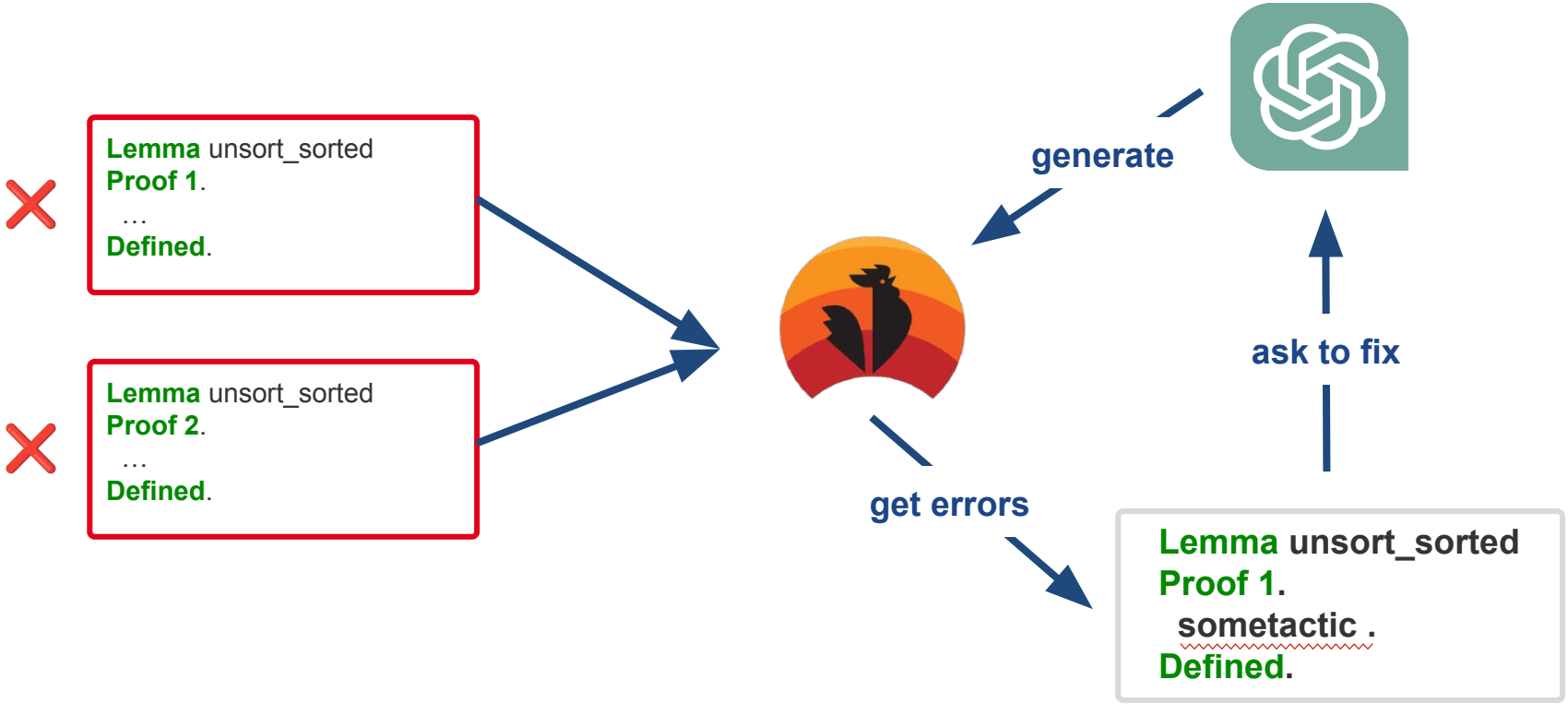
Proof repair



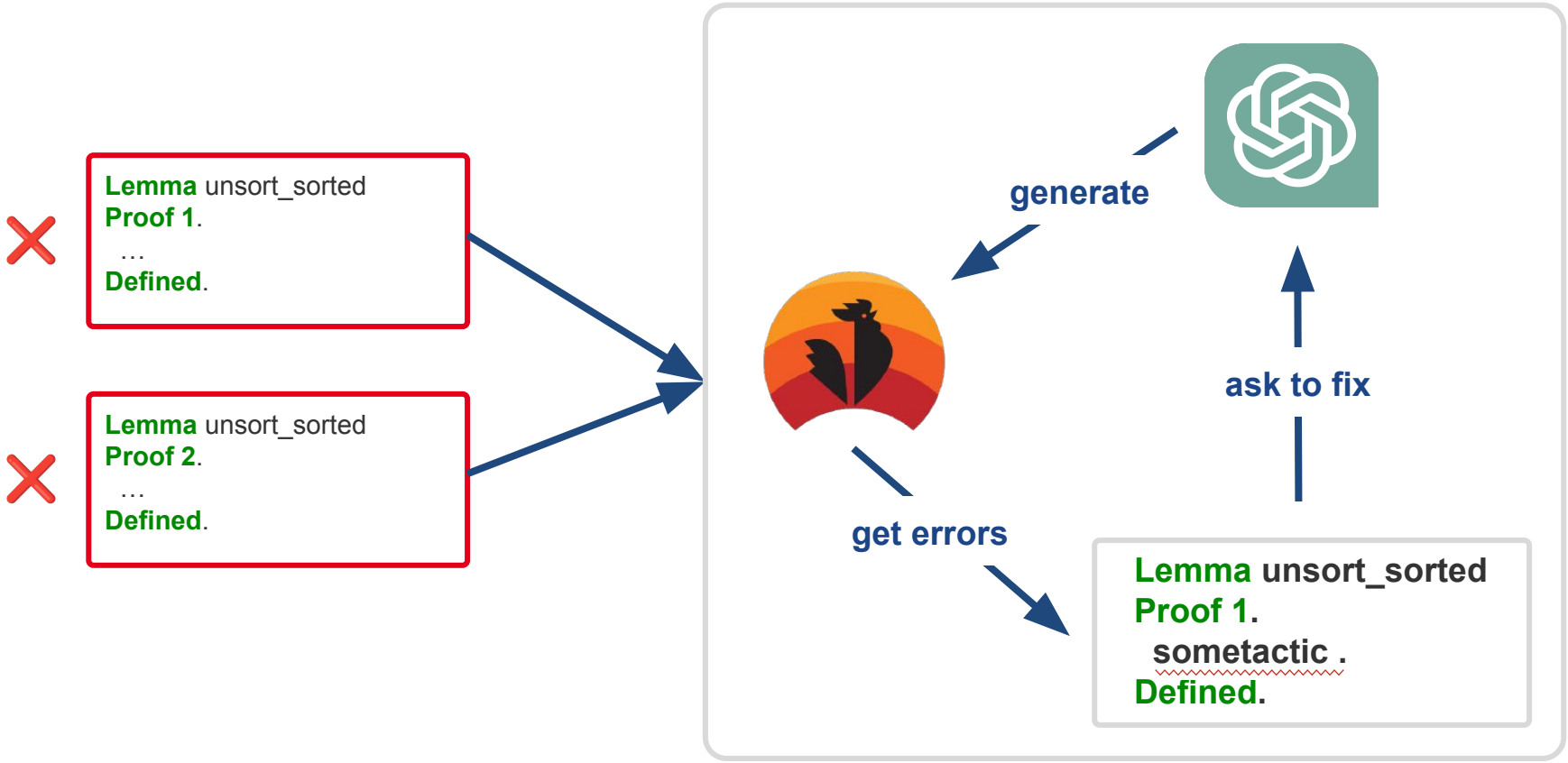
Proof repair



Proof repair



Proof repair



Research Questions

Research Questions

- **RQ1**: How well general purpose LLMs can write Coq proofs?

Research Questions

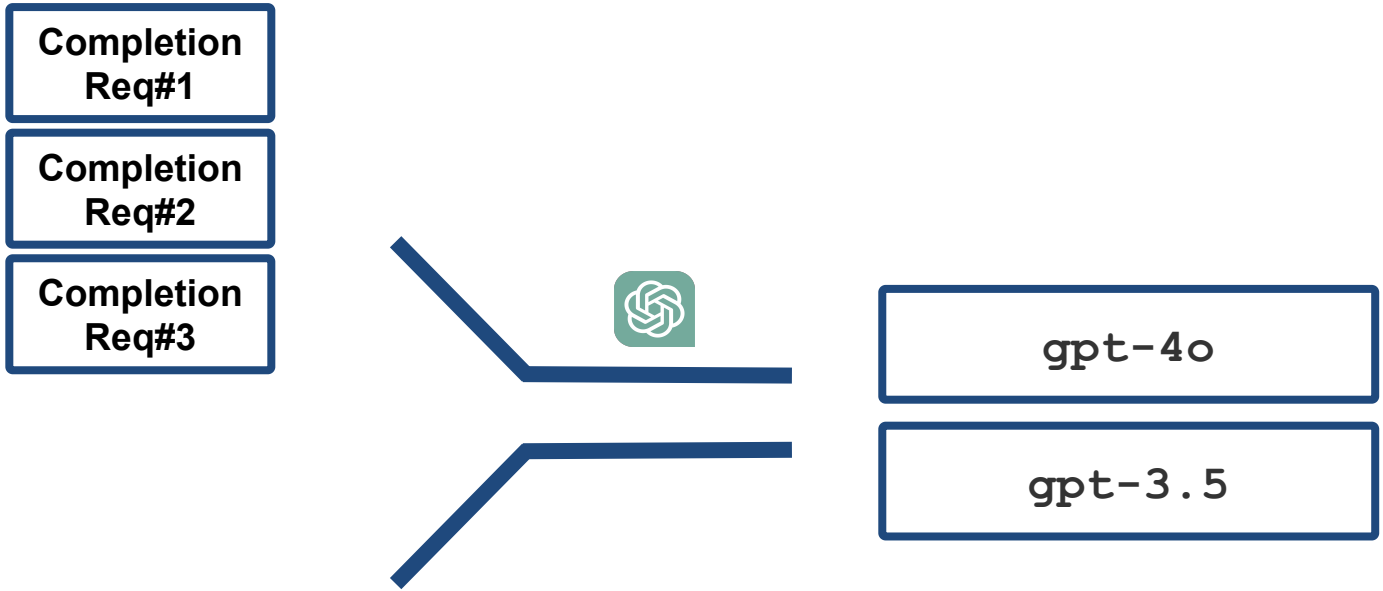
- **RQ1**: How well general purpose LLMs can write Coq proofs?
- **RQ2**: To which extent does CoqPilot improve the LLM approach to Coq generation?

Research Questions

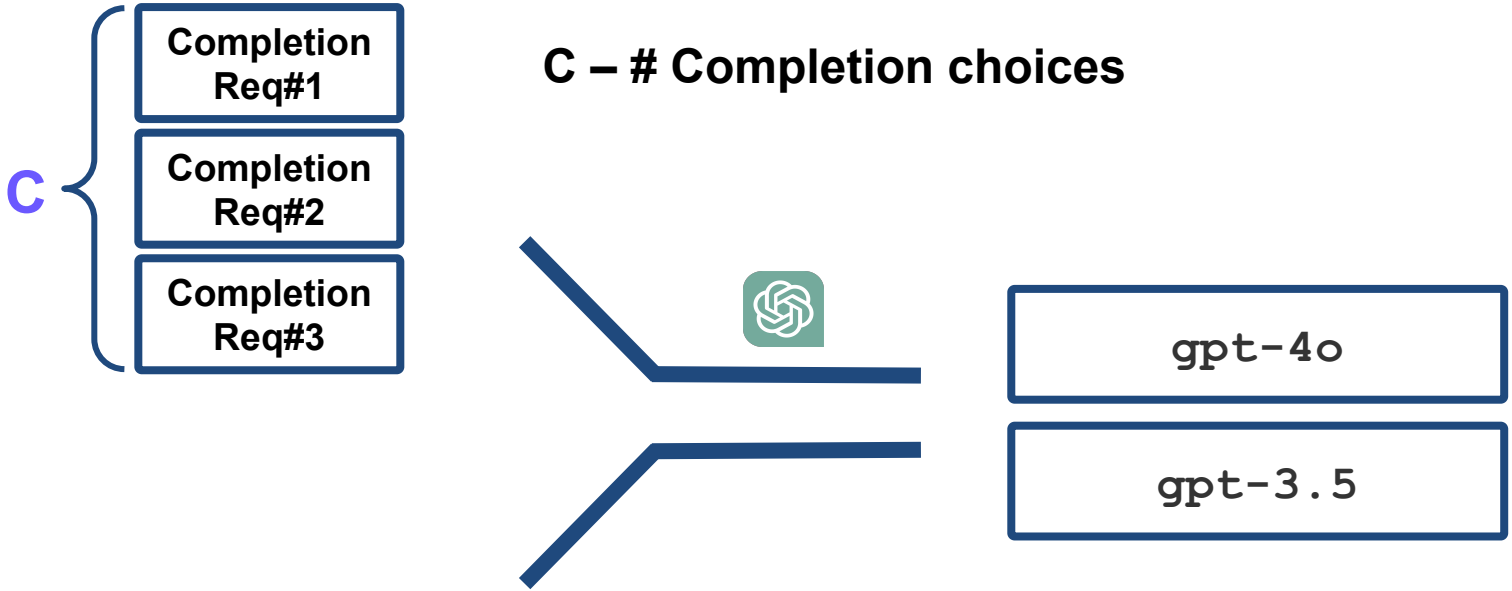
- **RQ1**: How well general purpose LLMs can write Coq proofs?
- **RQ2**: To which extent does CoqPilot improve the LLM approach to Coq generation?
- **RQ3**: What is the additional value CoqPilot contributes to other Coq automation tools such as CoqHammer and Tactician?

Benchmarking Framework

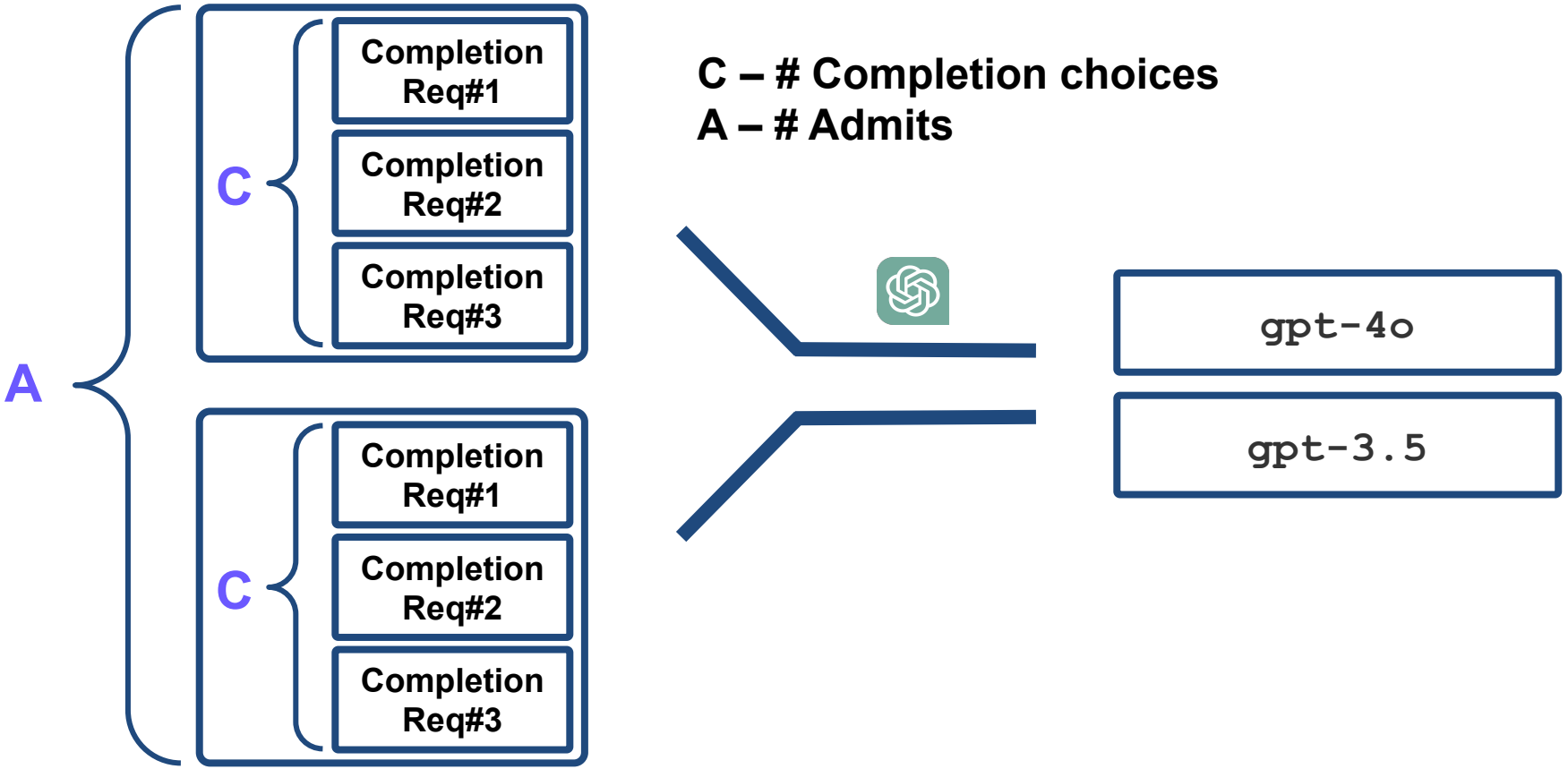
Benchmarking Framework



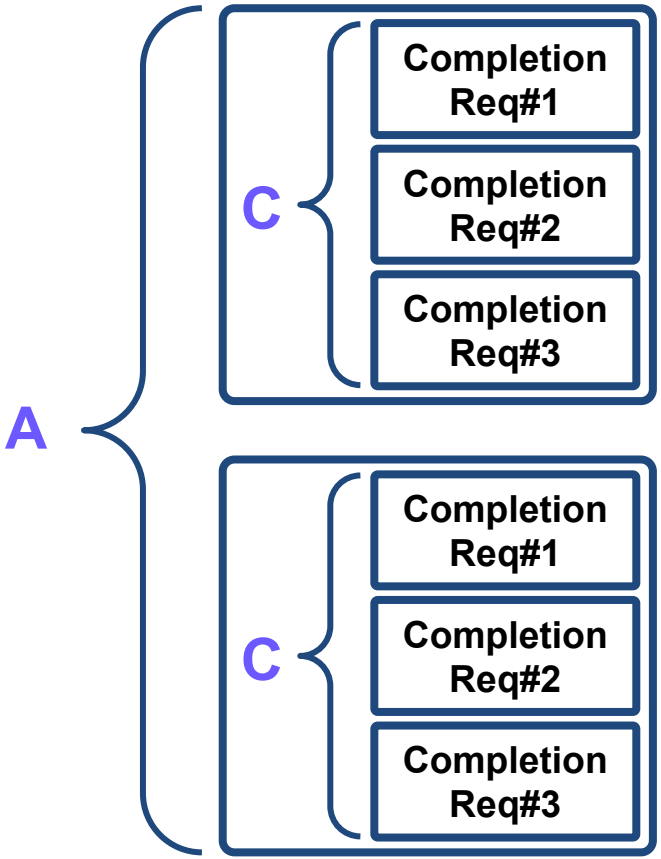
Benchmarking Framework



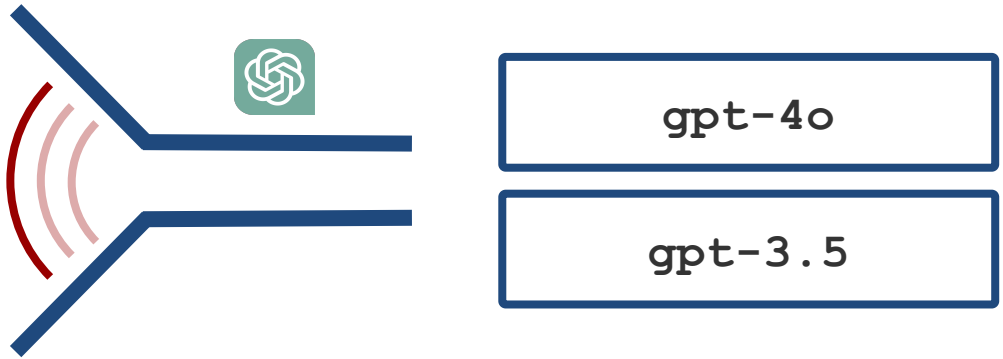
Benchmarking Framework



Benchmarking Framework

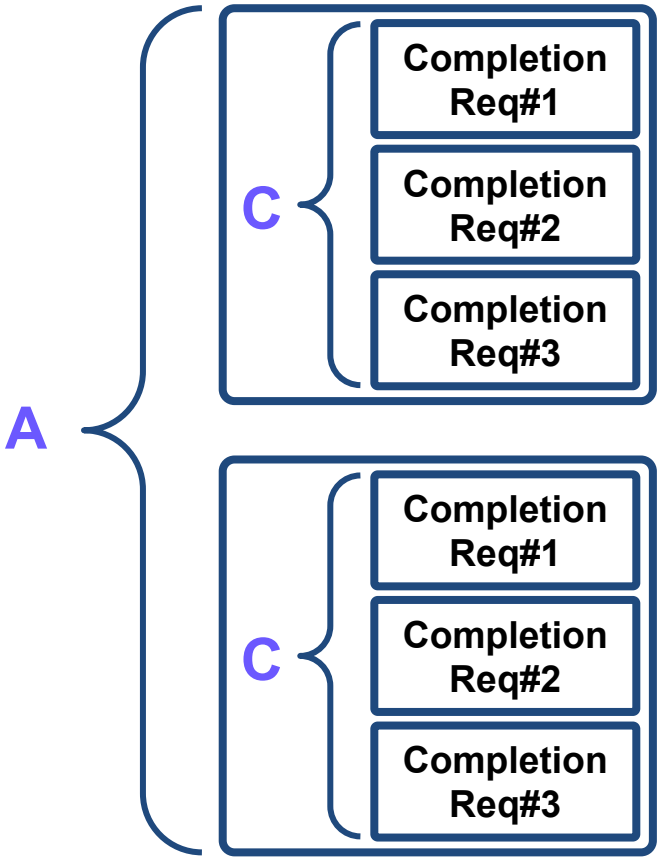


C – # Completion choices
A – # Admits

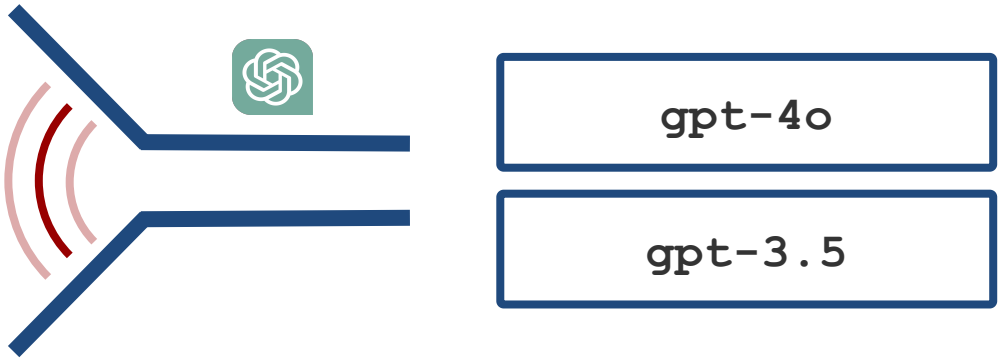


1. **Contextual limits**
2. Token rate limits
3. Request rate limits

Benchmarking Framework

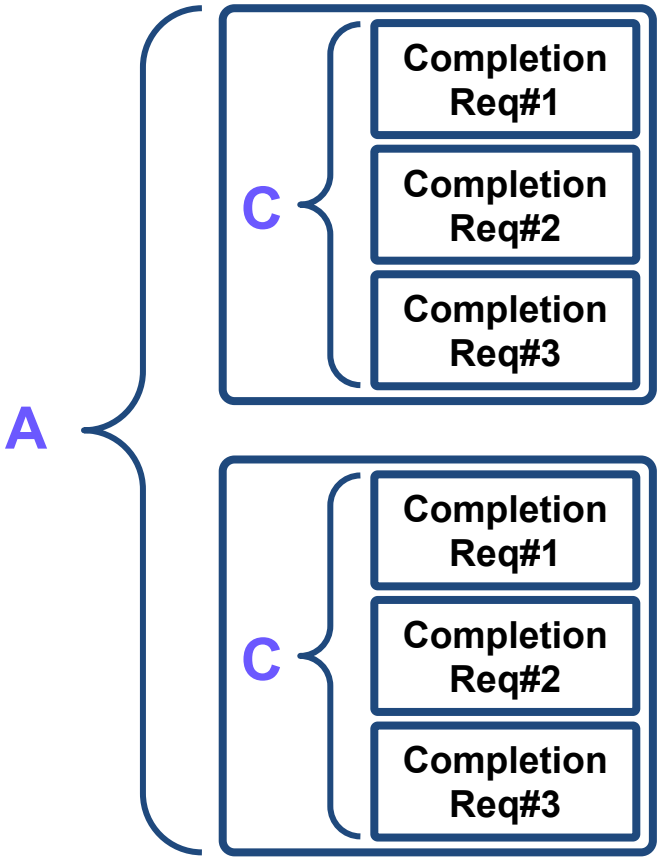


C – # Completion choices
A – # Admits

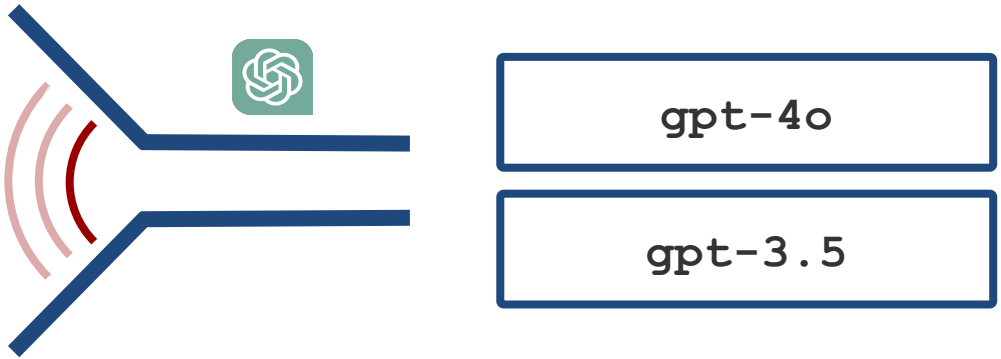


1. Contextual limits
2. **Token rate limits**
3. Request rate limits

Benchmarking Framework

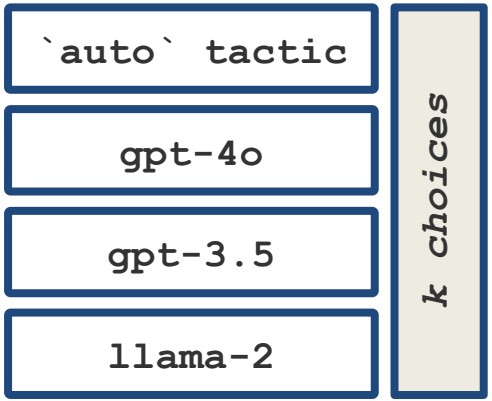


C – # Completion choices
A – # Admits



1. Contextual limits
2. Token rate limits
3. **Request rate limits**

Benchmarking Framework



Results

In the table below you can find the results of our experiments. For each of the theorems we show its group and the methods which proved the theorem during our experiments.

File	Theorem Name	Predefined tactic	OpenAI GPT-4	OpenAI GPT-3.5	LLaMA-2 13B Chat
basic/Execution_eco.v	`rf_rmw_ct_in_co`	×	✓	×	×
imm/imm_hb.v	`cr_hb_hb`	×	✓	✓	×
basic/FinExecution.v	`fin_exec_same_events`	✓	✓	✓	×
basic/Execution.v	`sb_trans`	×	×	×	×
basic/Execution.v	`sb_same_loc_W_trans`	×	✓	×	×
basic/Events.v	`restr_eq_rel_same_loc`	✓	✓	✓	✓
basic/Events.v	`same_loc_loceq`	✓	✓	✓	×

Benchmarking Framework

```
interface Benchmark {
    name: string;
    items: DatasetItem[];
    inputModelsParams: InputModelsParams;
    requireAllAdmitsCompleted: Boolean;
    benchmarkFullTheorems: Boolean;
    benchmarkAdmits: Boolean;
    timeoutMinutes: number;
    groupName: string;
    additionalImports?: AdditionalFileImport[];
    maximumUsedPremisesAmount?: number;
    perProofTimeoutMillis: number;
}
```

Evaluation

Evaluation

Reference proof length	⩽ 4	5 – 8	9 – 20	Total
Group size	131	98	71	300
`firstorder auto with *`	11%	2%	1%	6%
OpenAI GPT-3.5	29%	17%	6%	20%
OpenAI GPT-4o	50%	26%	15%	34%
LLaMA-2 13B Chat	2%	0%	0%	0.5%
Anthropic Claude	21%	7%	7%	13%
All models together	57%	32%	18%	39%
Tactician	45%	23%	10%	29%
CoqHammer	23%	4%	0%	11%
All methods together	71%	45%	23%	51%

IMM Project: https://github.com/weakmemory/imm
CoqPilot ASE'24 Tool Demo paper on: https://podkopaev.net/

Evaluation

Reference proof length	⩽ 4	5 – 8	9 – 20	Total
Group size	131	98	71	300
`firstorder auto with *`	11%	2%	1%	6%
OpenAI GPT-3.5	29%	17%	6%	20%
OpenAI GPT-4o	50%	26%	15%	34%
LLaMA-2 13B Chat	2%	0%	0%	0.5%
Anthropic Claude	21%	7%	7%	13%
All models together	57%	32%	18%	39%
Tactician	45%	23%	10%	29%
CoqHammer	23%	4%	0%	11%
All methods together	71%	45%	23%	51%

IMM Project: https://github.com/weakmemory/imm
CoqPilot ASE'24 Tool Demo paper on: https://podkopaev.net/

Evaluation

Reference proof length	⩽ 4	5 – 8	9 – 20	Total
Group size	131	98	71	300
`firstorder auto with *`	11%	2%	1%	6%
OpenAI GPT-3.5	29%	17%	6%	20%
OpenAI GPT-4o	50%	26%	15%	34%
LLaMA-2 13B Chat	2%	0%	0%	0.5%
Anthropic Claude	21%	7%	7%	13%
All models together	57%	32%	18%	39%
Tactician	45%	23%	10%	29%
CoqHammer	23%	4%	0%	11%
All methods together	71%	45%	23%	51%

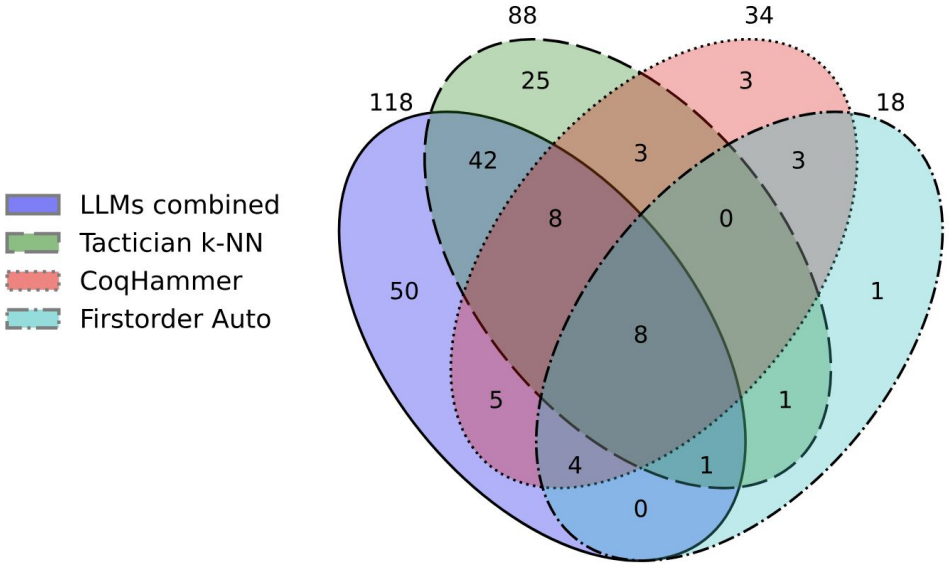
IMM Project: https://github.com/weakmemory/imm
CoqPilot ASE'24 Tool Demo paper on: https://podkopaev.net/

Evaluation

Reference proof length	$\leqslant 4$	$5-8$	$9-20$	Total
Group size	131	98	71	300
`firstorder auto with *`	11%	2%	1%	6%
OpenAI GPT-3.5	29%	17%	6%	20%
OpenAI GPT-4o	50%	26%	15%	34%
LLaMA-2 13B Chat	2%	0%	0%	0.5%
Anthropic Claude	21%	7%	7%	13%
All models together	57%	32%	18%	39%
Tactician	45%	23%	10%	29%
CoqHammer	23%	4%	0%	11%
All methods together	71%	45%	23%	51%

IMM Project: https://github.com/weakmemory/imm
CoqPilot ASE'24 Tool Demo paper on: https://podkopaev.net/

Evaluation



IMM Project: https://github.com/weakmemory/imm
CoqPilot ASE'24 Tool Demo paper on: https://podkopaev.net/

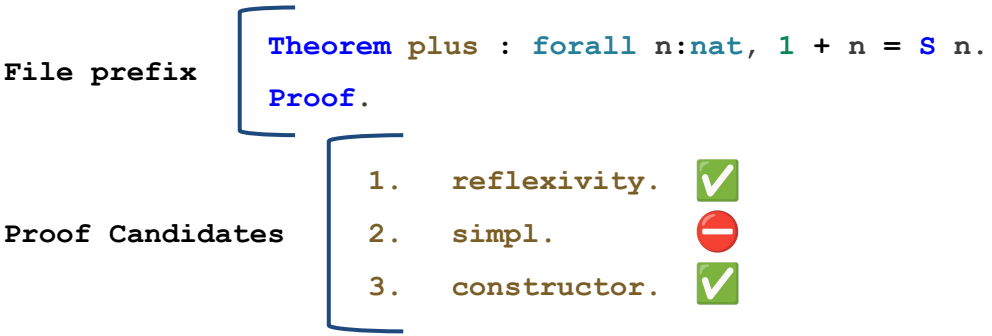
Technical challenges: Coq-LSP

Technical challenges: Coq-LSP

```
Theorem plus : forall n:nat, 1 + n = S n.
Proof.
   admit.
Admitted.
```

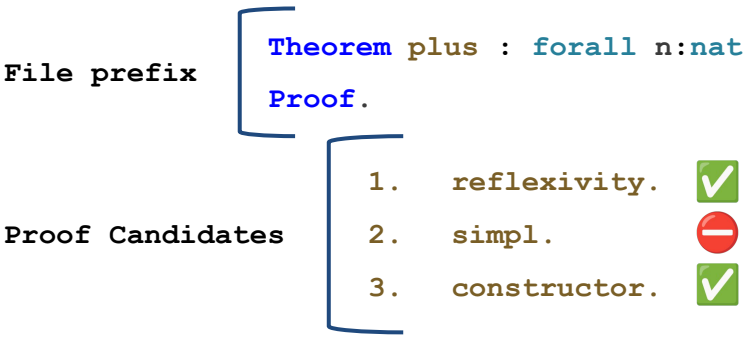
Technical challenges: Coq-LSP

```
Theorem plus : forall n:nat, 1 + n = S n.
Proof.
   admit.
Admitted.
```



Technical challenges: Coq-LSP

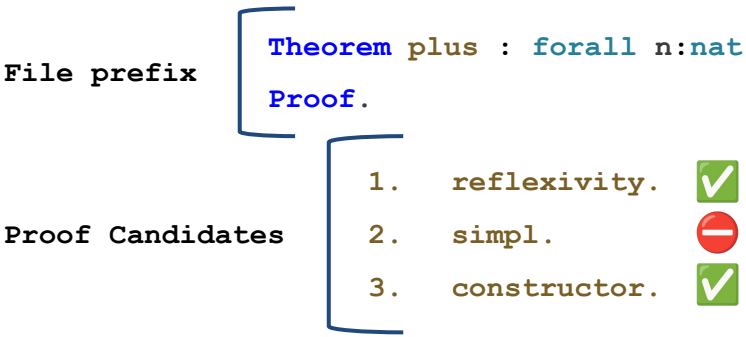
```
Theorem plus : forall n:nat, 1 + n = S n.
Proof.
   admit.
Admitted.
```




```
createTempFile()
file.write(prefix)
lsp.openDocument()
for proof in proofCandidates:
      file.append(proof)
      lsp.checkFile()
      file.remove(proof)
```

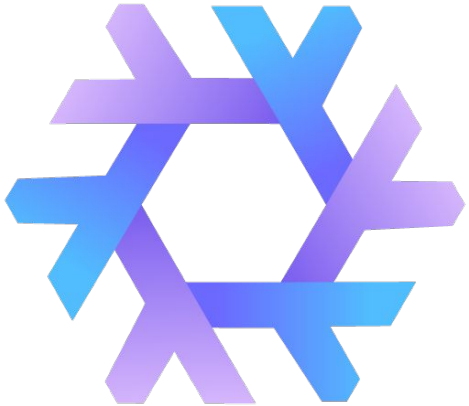
Technical challenges: Coq-LSP

```
Theorem plus : forall n:nat, 1 + n = S n.
Proof.
   admit.
Admitted.
```

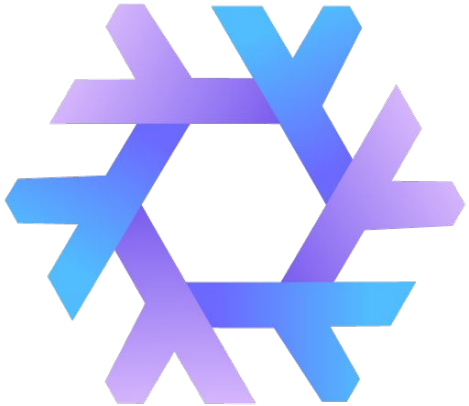


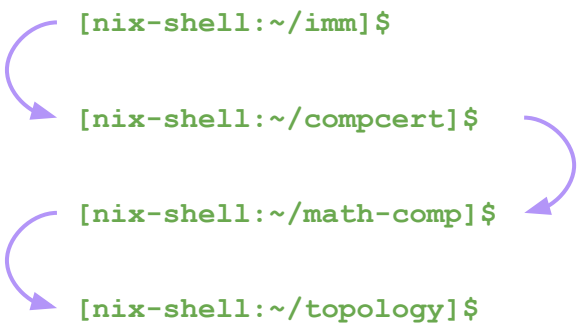

```
createTempFile()
file.write(prefix)
lsp.openDocument()
for proof in proofCandidates:
      file.append(proof)
      lsp.checkFile()
      file.remove(proof)
```

Technical challenges: Nix (1/2)

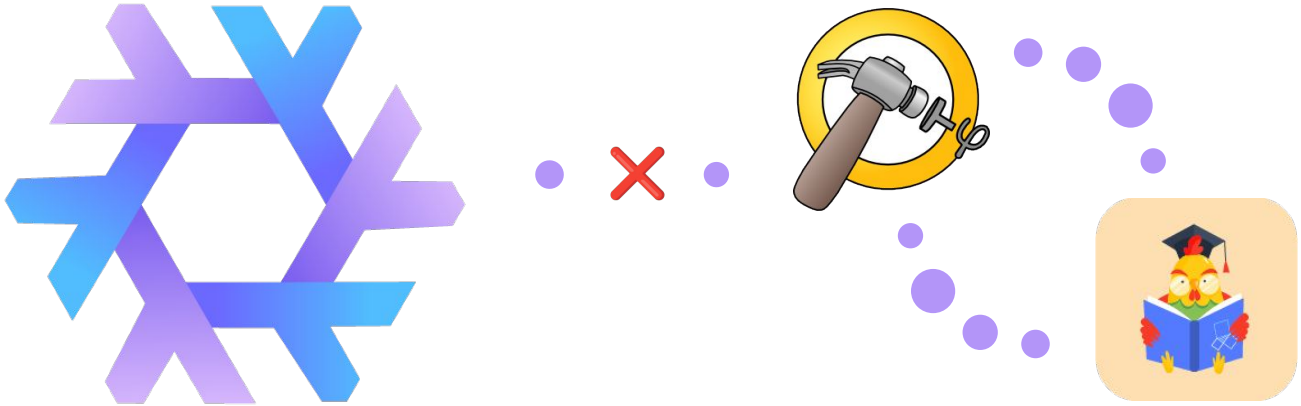


Technical challenges: Nix (1/2)





Technical challenges: Nix (2/2)



Improvement directions

Improvement directions

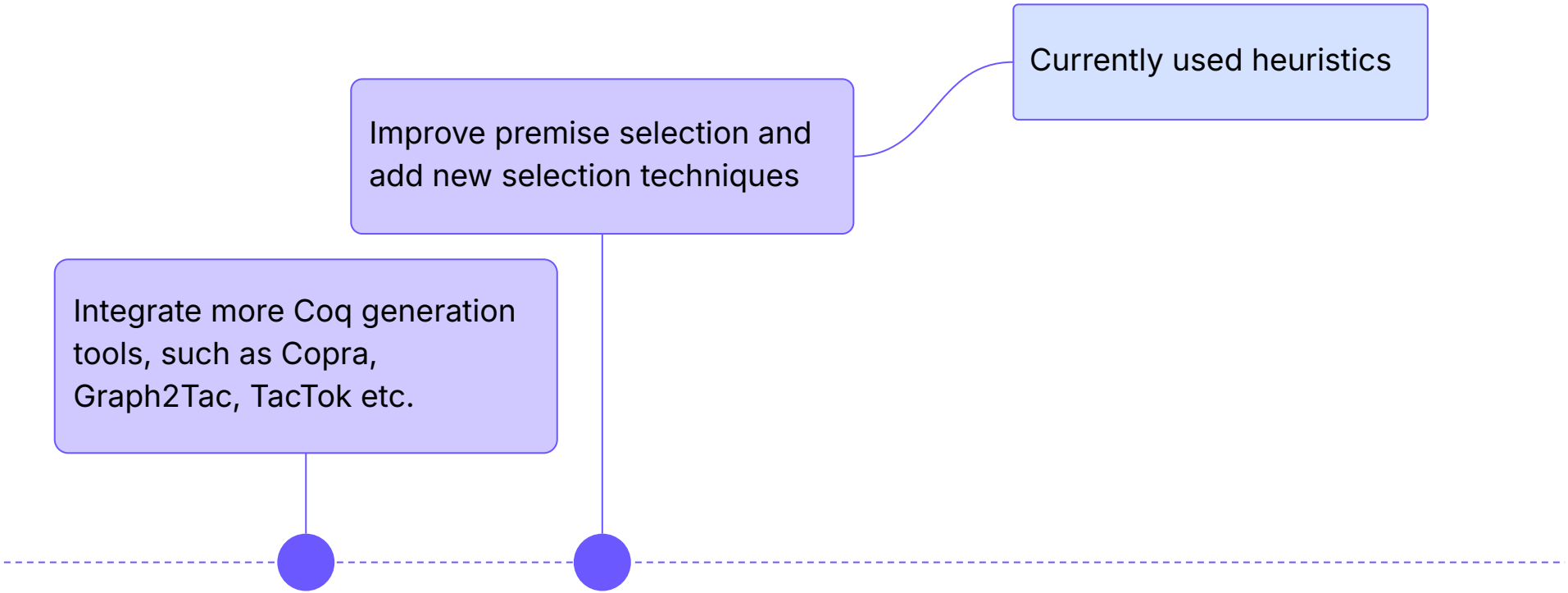
Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Improvement directions

Improve premise selection and add new selection techniques

Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Improvement directions



Improvement directions

Improve premise selection and add new selection techniques

Currently used heuristics

Distance from the target theorem

Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Improvement directions

Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Improve premise selection and add new selection techniques

Currently used heuristics

Distance from the target theorem

Similarity of the goal

Improvement directions

Improve premise selection and add new selection techniques

Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Explore and improve locally available models in order to make inference cheaper and preserve privacy

Improvement directions

Improve premise selection and add new selection techniques

Integrate more Coq generation tools, such as Copra, Graph2Tac, TacTok etc.

Please talk to us if you have ideas!

CoqPilot: a plugin for LLM-based generation of proofs

JetBrains-Research/**coqpilot**

extension: **coqpilot**

{andrei.kozyrev, gleb.solovev, nikita.khramov, anton.podkopaev}@jetbrains.com

Programming Languages and Program Analysis Lab (PLAN), JetBrains Research