Can LLMs Verify System Software?

A Case Study Using FSCQ as a Benchmark

Jianxing Qin, Alexander Du, Danfeng Zhang, Matthew Lentz, Danyang Zhuo

Duke
UNIVERSITY

System software verification is labor-intensive

Verification requires **many times** more proof than code

System	Lines of Code	Lines of Proof	Human Effort
seL4 (2009)	~10k	~200k	22py
CertiKOS (2016)	~6k	~50k	2py
CompCert (2008)	~6k	~36k	3py
DFSCQ (2017)	~12k	~72k	-

Can LLMs **augment or replace** the manual verification of system software?

Existing Work

Mathematics 

- Informal theorem proving
- Formal theorem proving
  - Tactic generation (GPT-f)
  - Proof search
  - Premise selection (LeanDojo)

Software Verification 

- Automated theorem proving
  - Invariant synthesis
- Interactive theorem proving
  - Small programs (FVEL)
  - System software (Selene, Rango)

System software poses unique challenges for LLMs

Real-world complexity

DFSCQ models concrete behaviors like disk I/O, crashes, and recovery

Customized logics

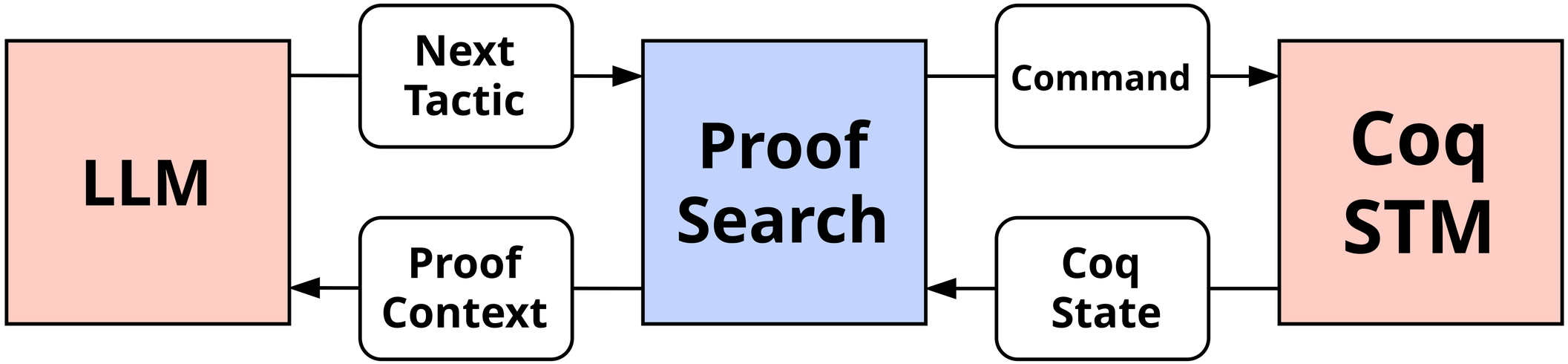
DFSCQ formalizes crash-consistency using Crash Hoare Logic (CHL)

Substantial project size

DFSCQ contains hundreds of theorems and lemmas across many files

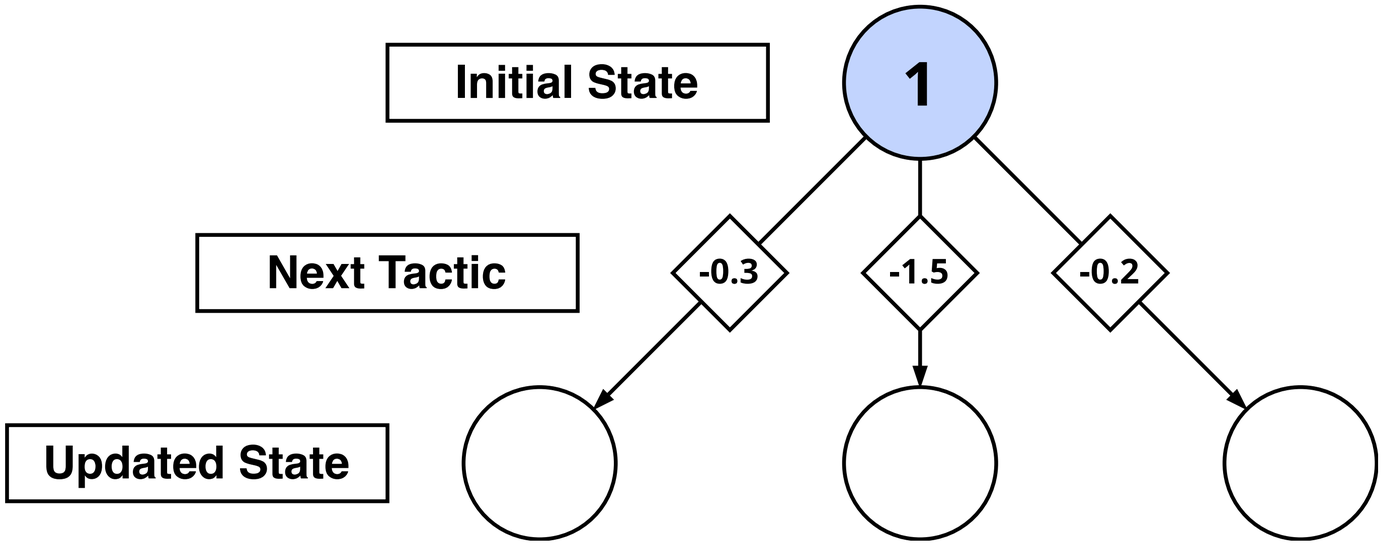
System Design

We built a system based on GPT-f and **augmented with proof context**

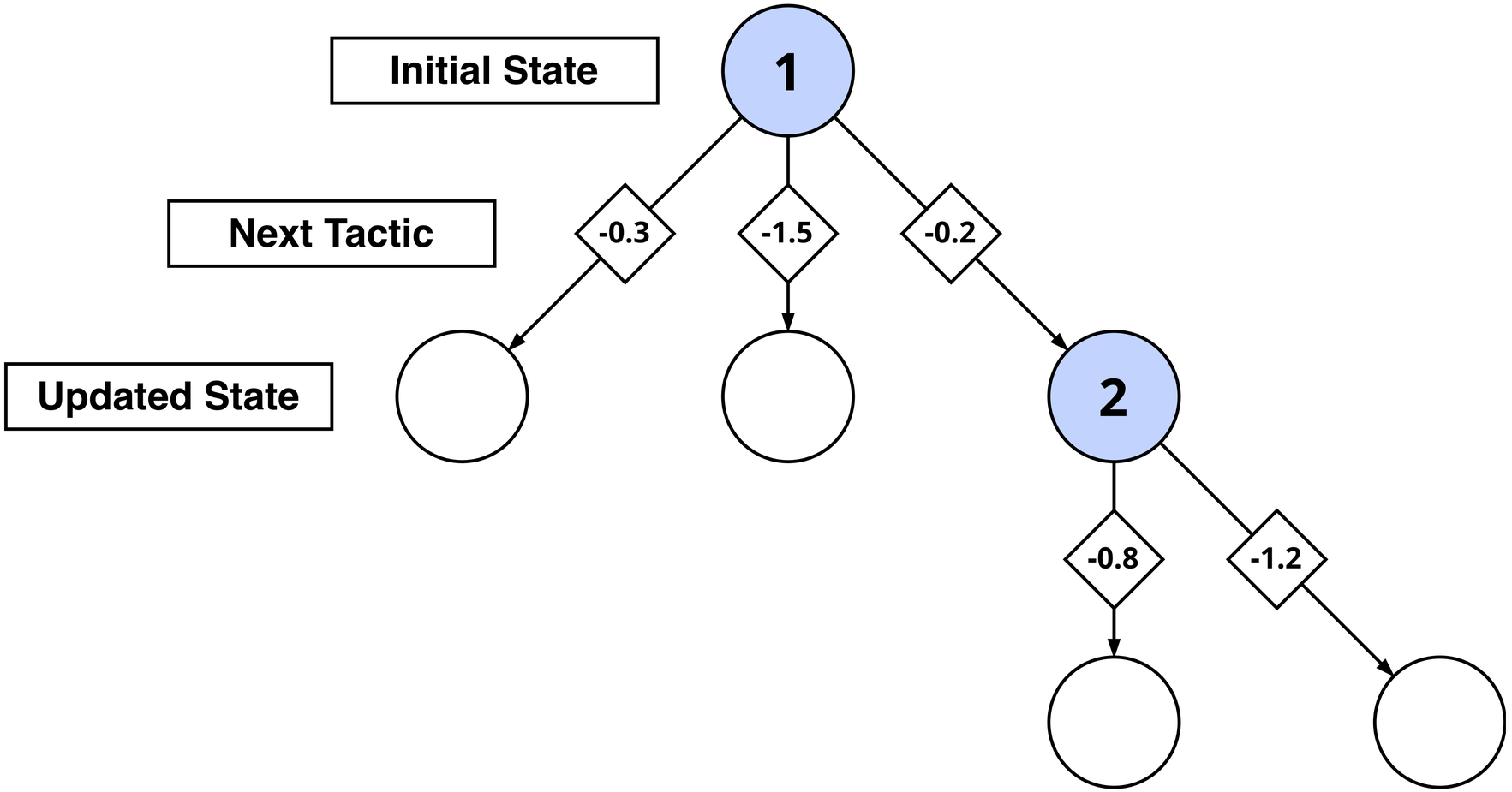


System Design: Proof Search (GPT-f)

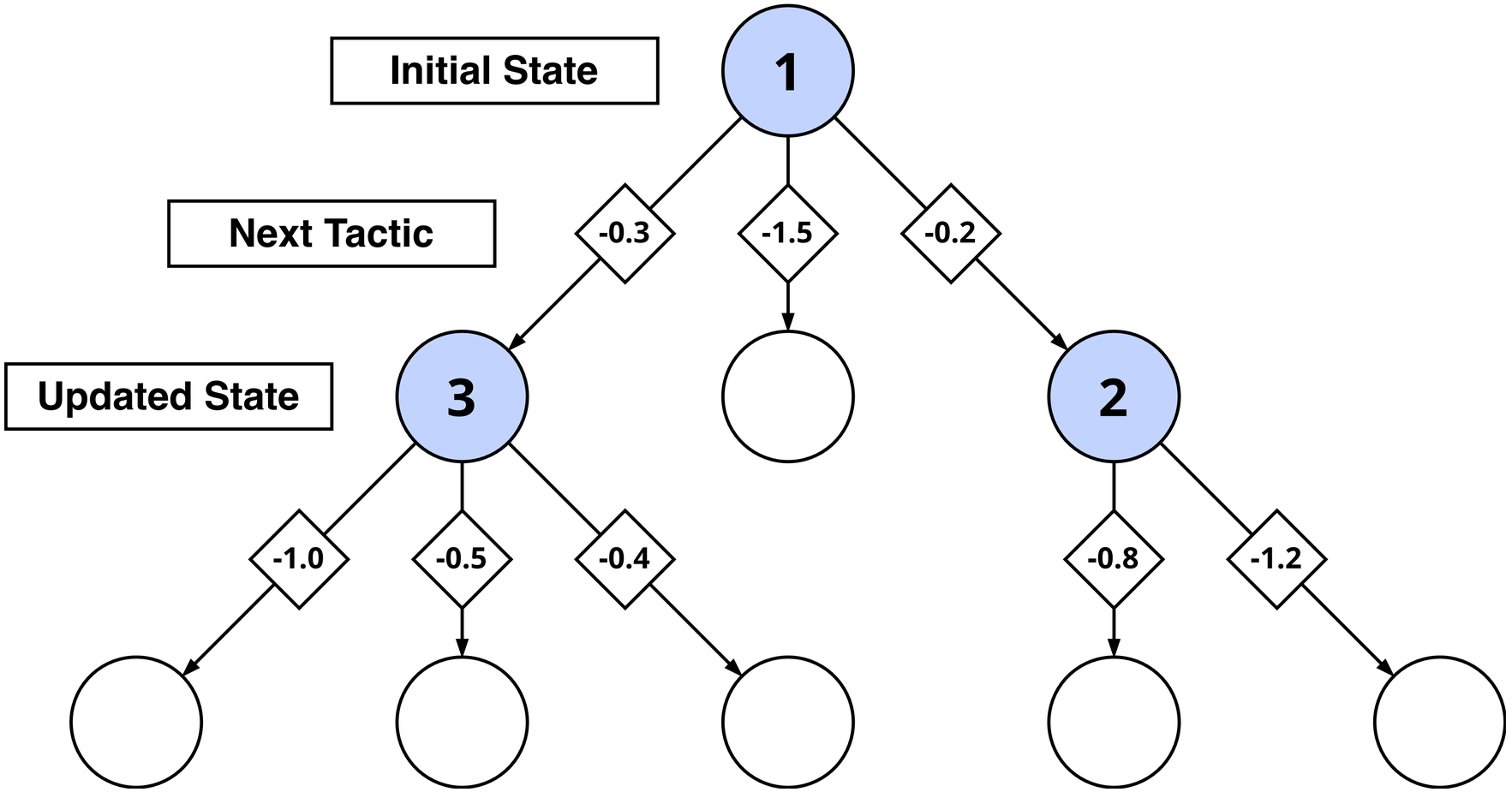
System Design: Proof Search (GPT-f)



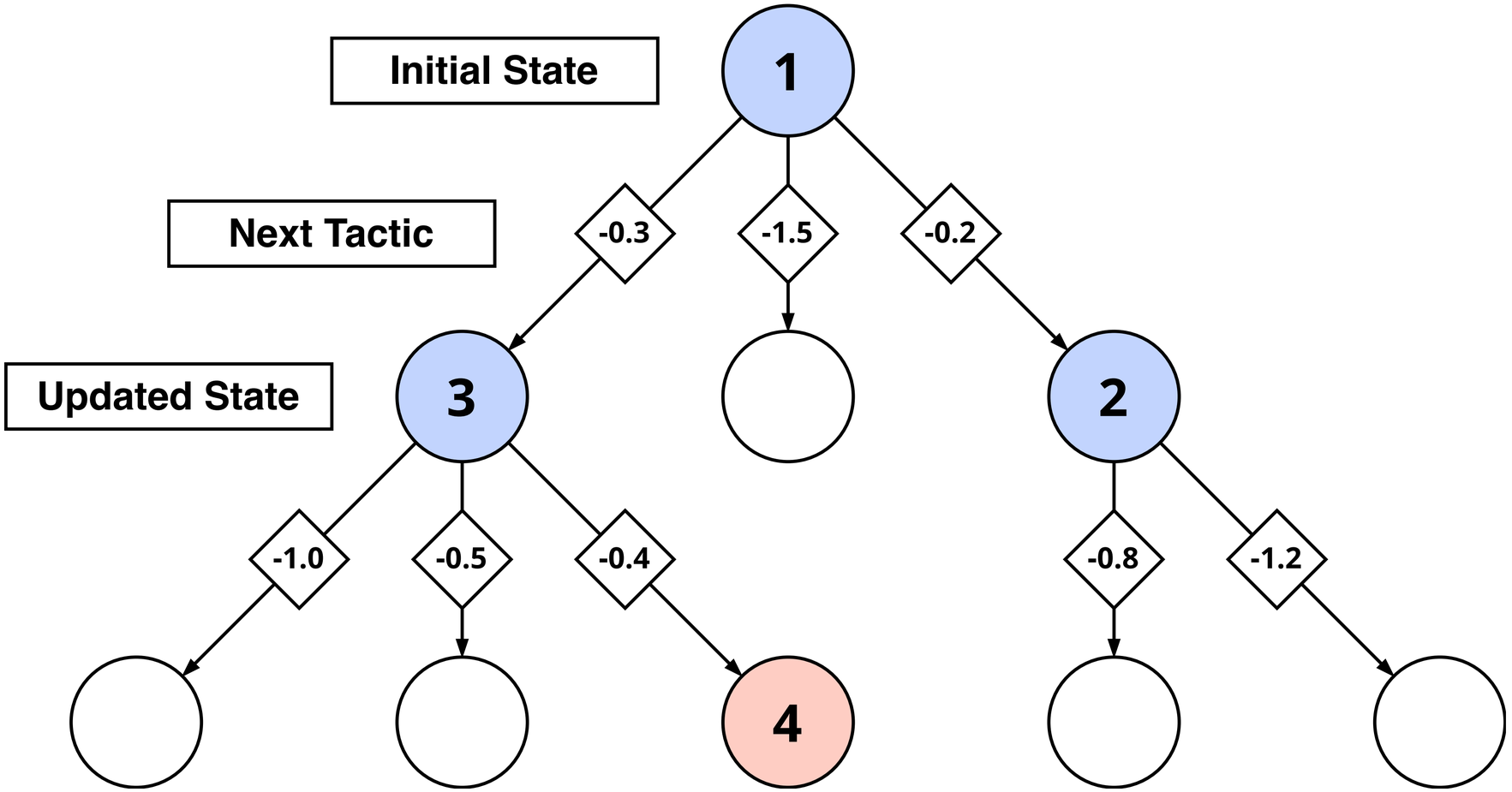
System Design: Proof Search (GPT-f)



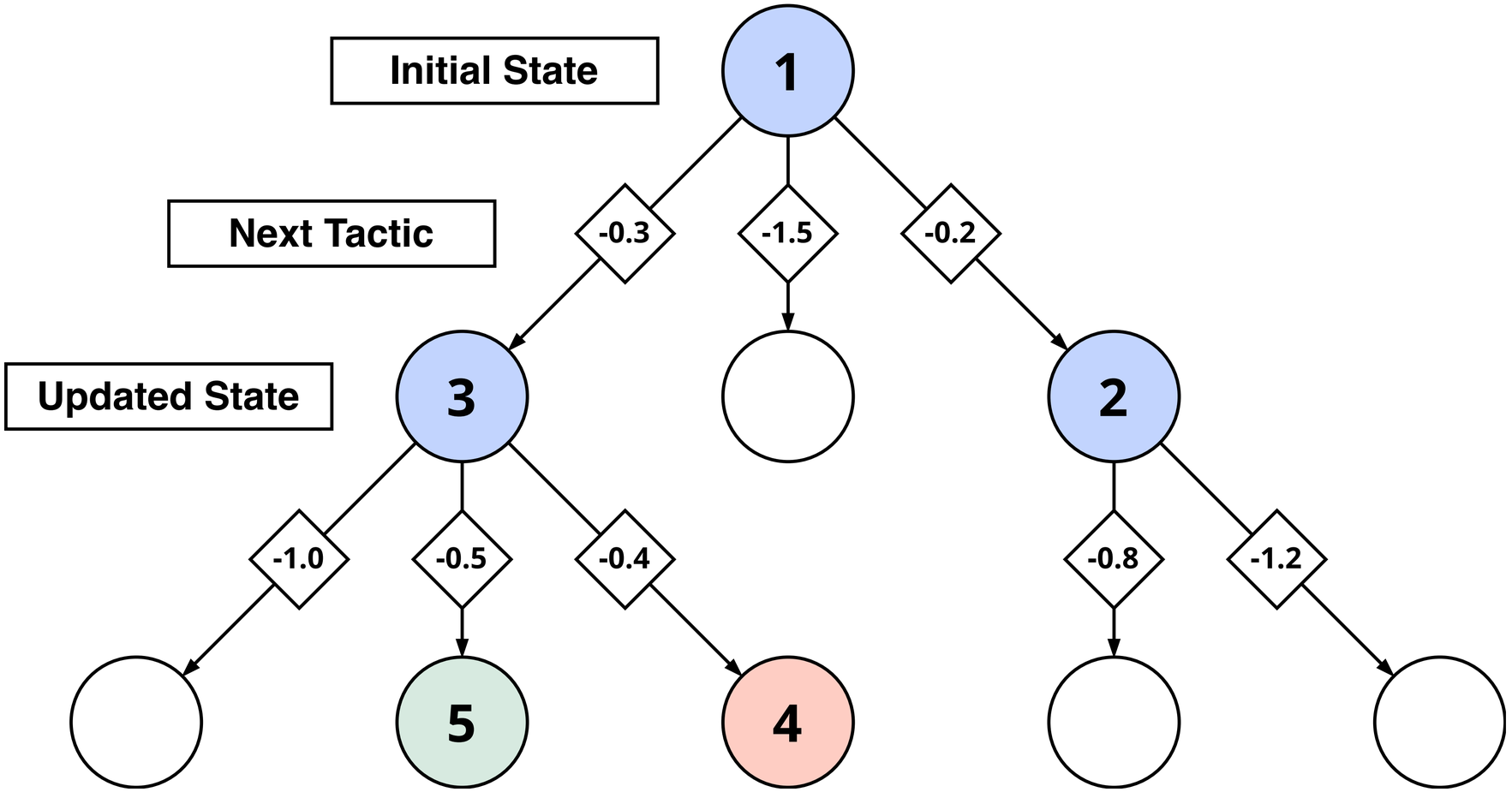
System Design: Proof Search (GPT-f)



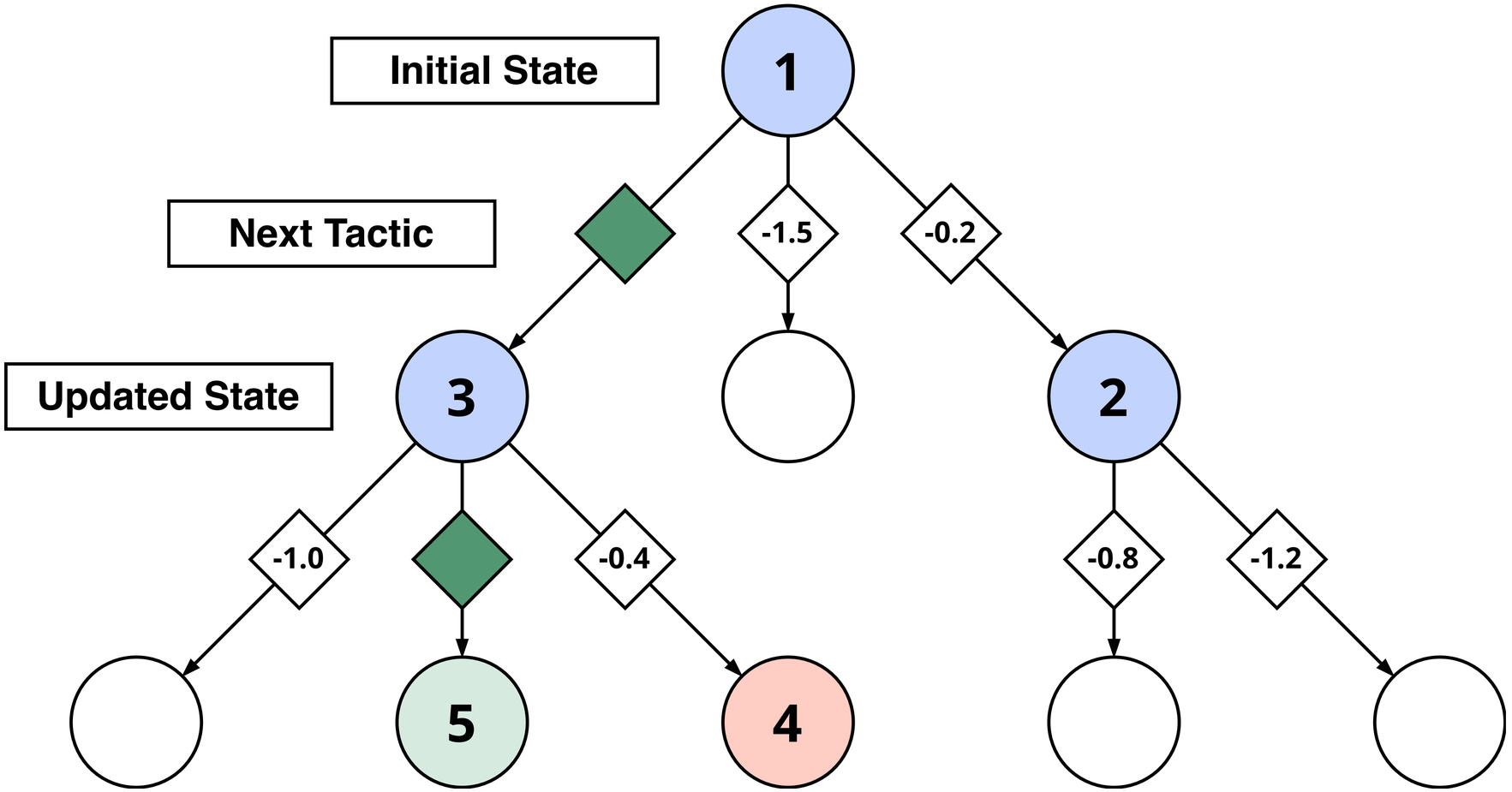
System Design: Proof Search (GPT-f)



System Design: Proof Search (GPT-f)



System Design: Proof Search (GPT-f)



System Design: Proof Context

Proofs may rely on **surrounding context**

```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intuition.
    eapply SDIR.crash_rep; eauto.
    inversion H; intuition subst; cbn in *.
    congruence.
  Qed.
```

DirCache.v

System Design: Proof Context

Proofs may rely on **surrounding context**

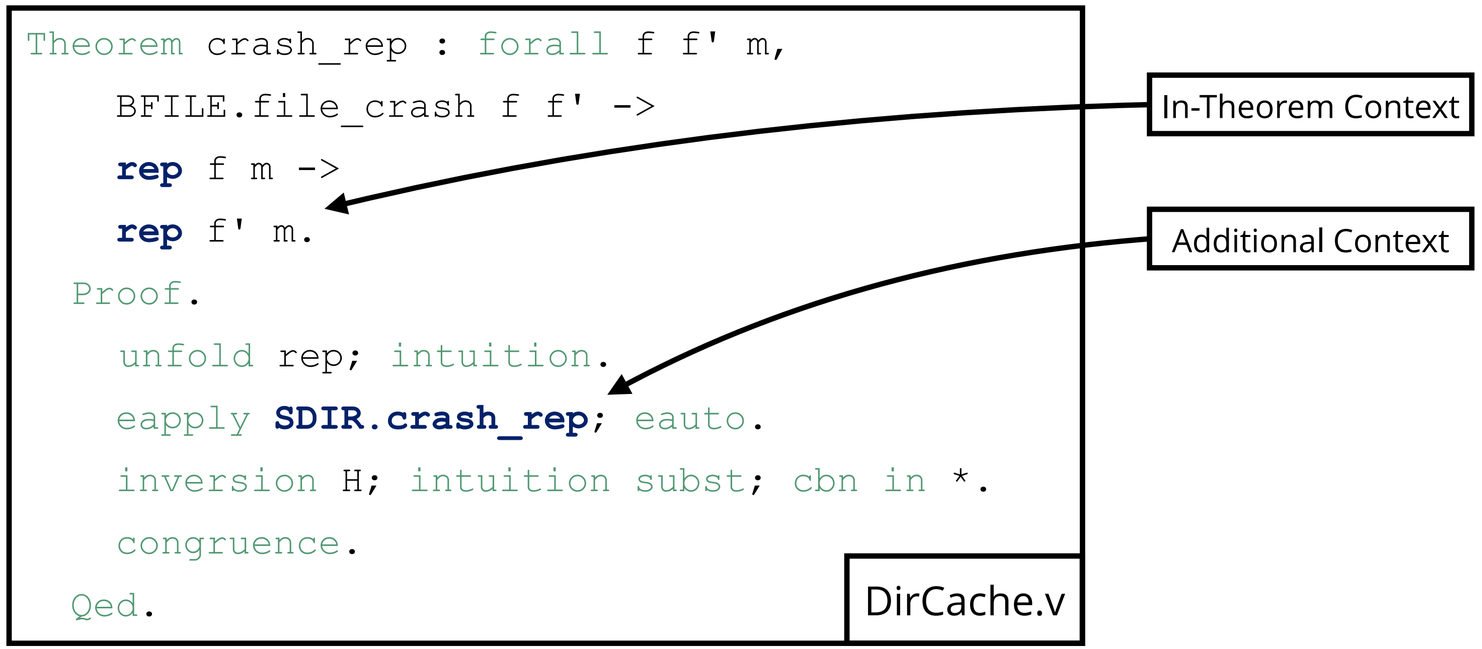
```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intuition.
    eapply SDIR.crash_rep; eauto.
    inversion H; intuition subst; cbn in *.
    congruence.
  Qed.
```

In-Theorem Context

DirCache.v

System Design: Proof Context

Proofs may rely on **surrounding context**



Provide **statements** of accessible definitions, lemmas, and theorems

System Design: Proof Context

Proofs may be **similar to other proofs**

```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intuition.
    eapply SDIR.crash_rep; eauto.
    inversion H; intuition subst; cbn in *.
    congruence.
  Qed.
```

DirCache.v

```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intros.
    repeat deex.
    eexists; intuition eauto.
    eapply DIR.crash_rep; eauto.
  Qed.
```

DirName.v

System Design: Proof Context

Proofs may be **similar to other proofs**

```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intuition.
    eapply SDIR.crash_rep; eauto.
    inversion H; intuition subst; cbn in *.
    congruence.
  Qed.
```

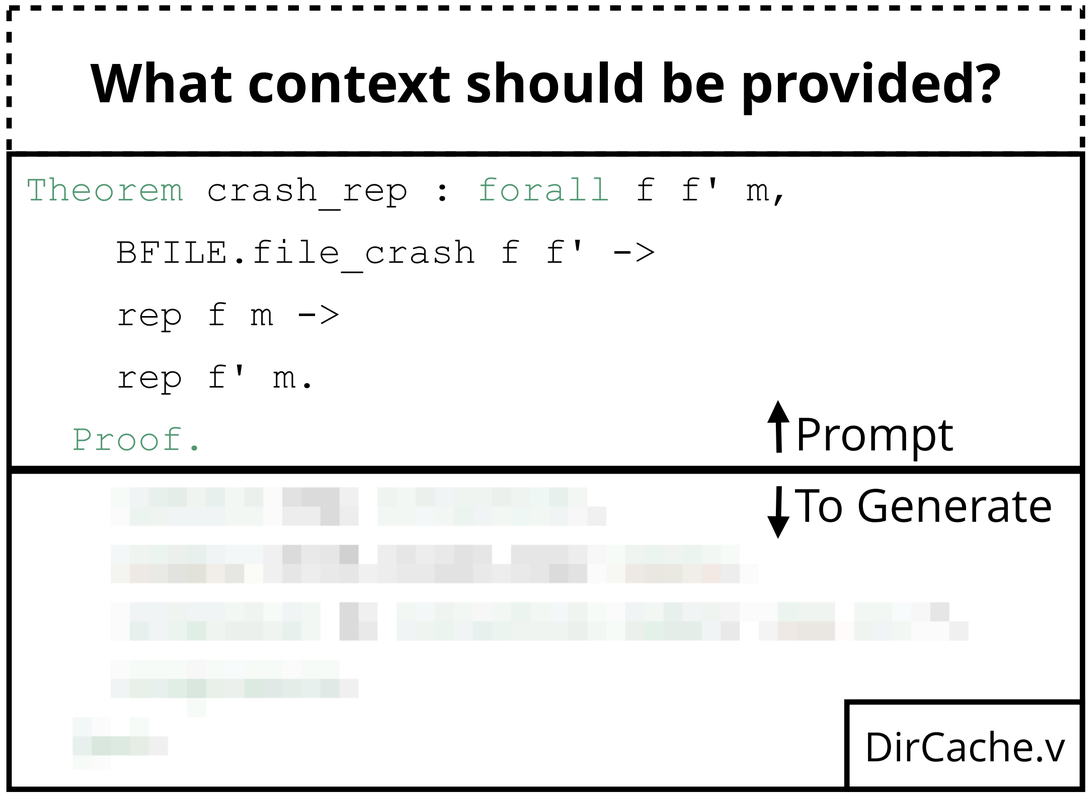
DirCache.v

```
Theorem crash_rep : forall f f' m,
    BFILE.file_crash f f' ->
    rep f m ->
    rep f' m.
  Proof.
    unfold rep; intros.
    repeat deex.
    eexists; intuition eauto.
    eapply DIR.crash_rep; eauto.
  Qed.
```

DirName.v

Provide **proofs** of lemmas and theorems **as “hints”**

System Design: Proof Context



Vanilla

Provide **statements** of all accessible definitions, theorems, and lemmas

Hint

Additionally provide **proofs** of 50% of accessible theorems and lemmas, selected at random

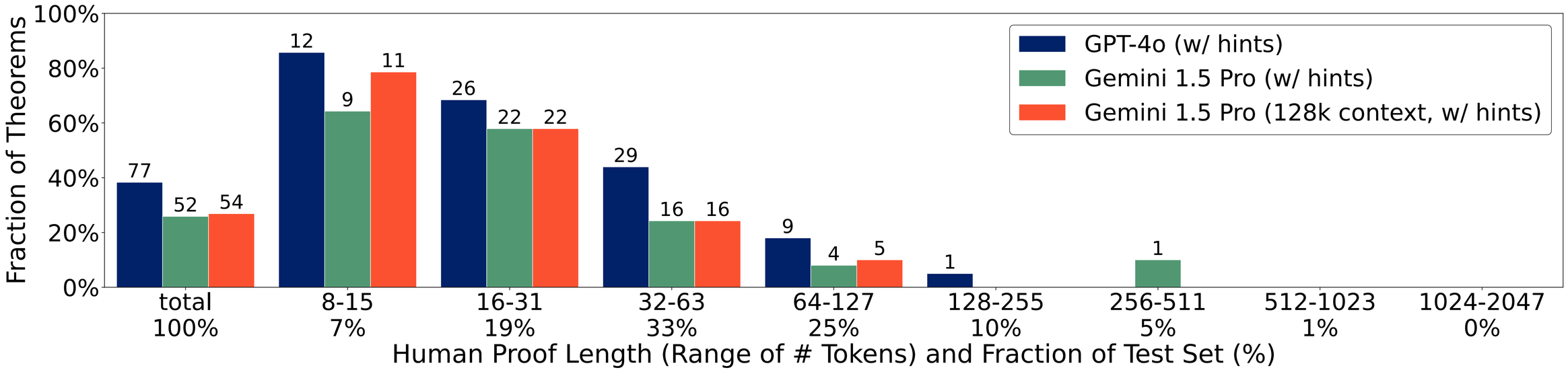
Evaluation

1. How much of FSCQ can be proved?
2. What is the effect of proof context?
3. Which theorems in FSCQ are easier (or harder) to prove?
4. Are models simply memorizing FSCQ's proofs?
5. What is the effect of model size?
6. When and why do models fail?

Evaluation

1. How much of FSCQ can be proved?
2. What is the effect of proof context?
3. Which theorems in FSCQ are easier (or harder) to prove?
4. Are models simply memorizing FSCQ's proofs?
5. What is the effect of model size?
6. When and why do models fail?

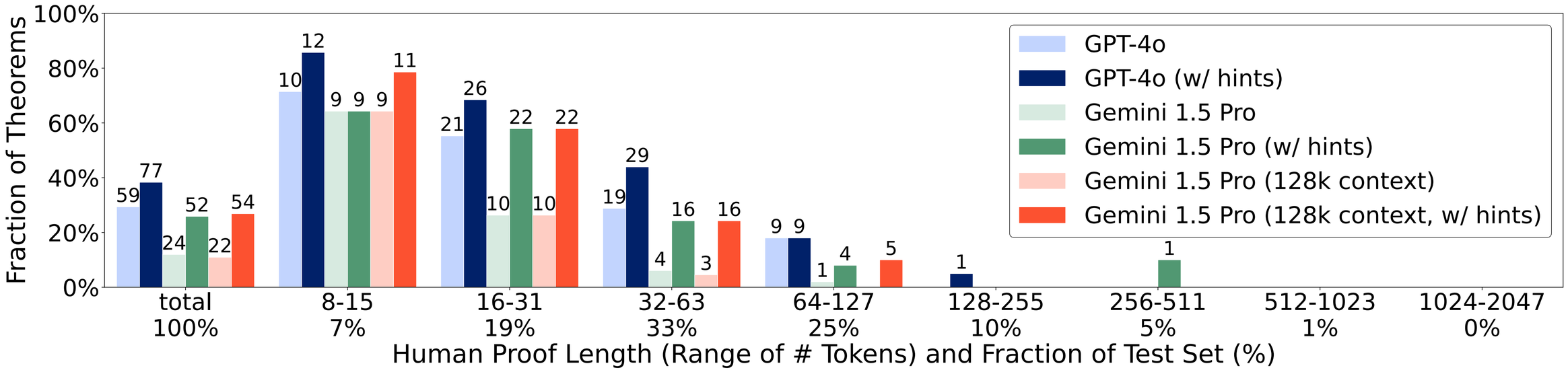
Evaluation: How much of FSCQ can be proved?



GPT-4o proves **38%** of the sampled theorems

...and **57%** of the sampled theorems under 64 tokens

Evaluation: What is the effect of proof context?



Hints improve coverage by between **30%** and **145%**

Longer context window **does not necessarily** improve coverage

Evaluation: Which theorems are easier to prove?

(With Hints)

	Actual	Expected
File System	20%	32%
CHL	52%	42%
Utilities	58%	47%

Based on lengths of human proofs and average proof coverage for the associated range of lengths

Evaluation: Which theorems are easier to prove?

(With Hints)

	Actual	Expected	
File System	20%	32%	
CHL	52%	42%	**+10%**
Utilities	58%	47%	**+11%**

Better than expected!

Based on lengths of human proofs and average proof completion rate for the associated range of # tokens

Evaluation: Which theorems are easier to prove?

(With Hints)

	Actual	Expected
File System	20%	32%
CHL	52%	42%
Utilities	58%	47%

-12% Worse than expected…

Based on lengths of human proofs and average proof completion rate for the associated range of # tokens

Evaluation: Which theorems are easier to prove?

	(With Hints)		(Vanilla)	
	Actual	**Expected**	**Actual**	**Expected**
File System	20%	32%	16%	24%
CHL	52%	42%	43%	32%
Utilities	58%	47%	40%	36%

Hints improved performance the most on **Utilities**

Evaluation: Are models memorizing FSCQ's proofs?

Generated proofs sometimes use lemmas **more efficiently**

```
Lemma ndata_log_padded_log : forall a, ndata_log (padded_log a) = ndata_log a.
```

```
unfold ndata_log, padded_log, setlen,
  roundup; intros.
rewrite firstn_oob by auto.
repeat rewrite map_app.
rewrite repeat_map; simpl.
rewrite nonzero_addrs_app.
setoid_rewrite <- app_nil_l at 3.
rewrite nonzero_addrs_app_zeros; auto.
```

Original

```
intros a.
unfold ndata_log.
rewrite nonzero_addrs_padded_log.
reflexivity.
```

GPT-4o

Evaluation: Are models memorizing FSCQ's proofs?

Generated proofs sometimes use lemmas **more efficiently**

```
Lemma ndata_log_padded_log : forall a, ndata_log (padded_log a) = ndata_log a.
```

```
unfold ndata_log, padded_log, setlen,
  roundup; intros.
rewrite firstn_oob by auto.
repeat rewrite map_app.
rewrite repeat_map; simpl.
rewrite nonzero_addrs_app.
setoid_rewrite <- app_nil_l at 3.
rewrite nonzero_addrs_app_zeros; auto.
```

Original

```
intros a.
unfold ndata_log.
rewrite nonzero_addrs_padded_log.
reflexivity.
```

Additional Context

GPT-4o

Evaluation: Are models memorizing FSCQ's proofs?

Generated proofs sometimes use lemmas **more efficiently**

```
Lemma tree_name_distinct_head: forall inum name l t,
  tree_names_distinct (TreeDir inum ((name, t)::l)) -> tree_names_distinct t.
```

Original

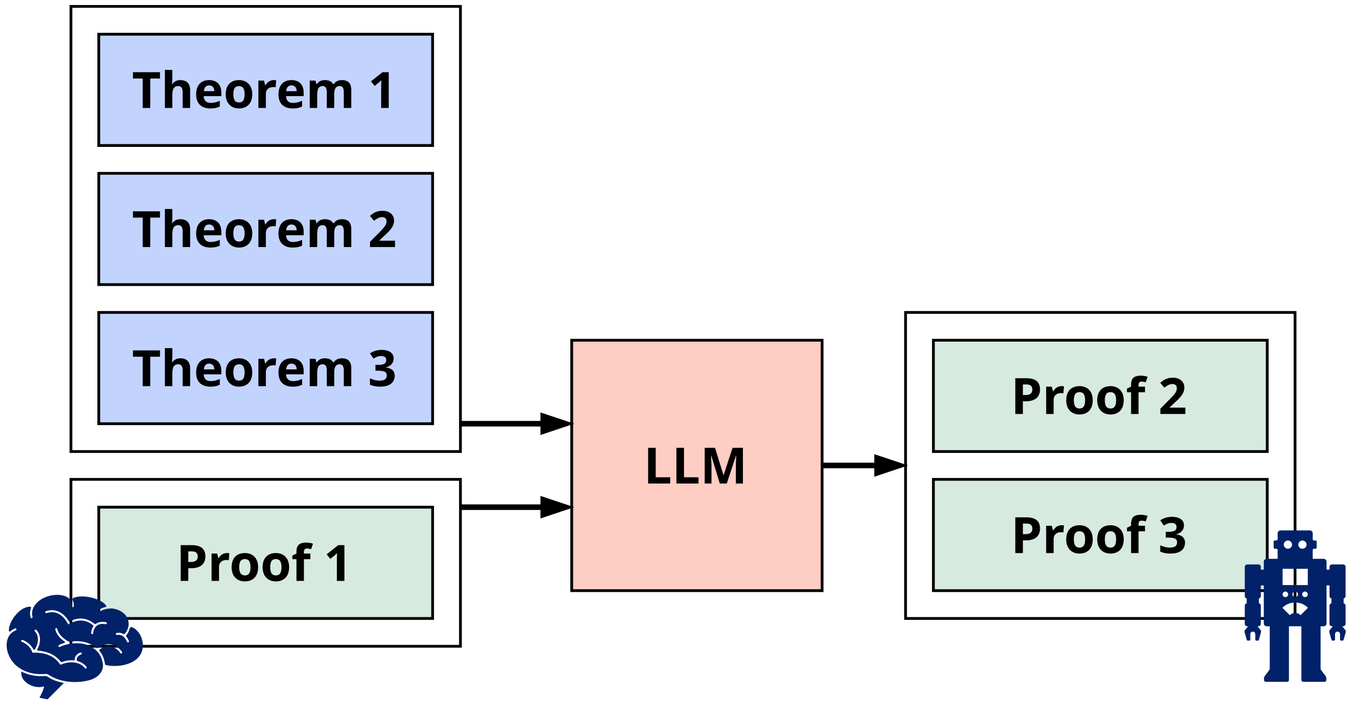
```
intros. destruct t.
constructor. inversion H.
rewrite map_cons in H2.
apply Forall_inv in H2.
simpl in H2. inversion H2.
constructor; eauto.
```

Gemini-1.5 Pro

```
intros.
inversion H; auto.
inversion H2; subst; auto.
```

Discussion: Augmenting human effort

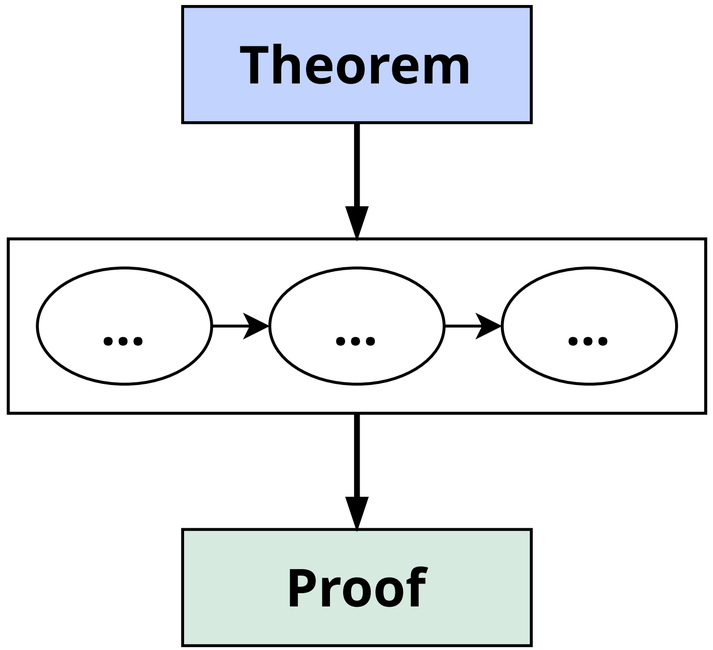
LLMs naturally fit into **human-in-the-loop** workflows

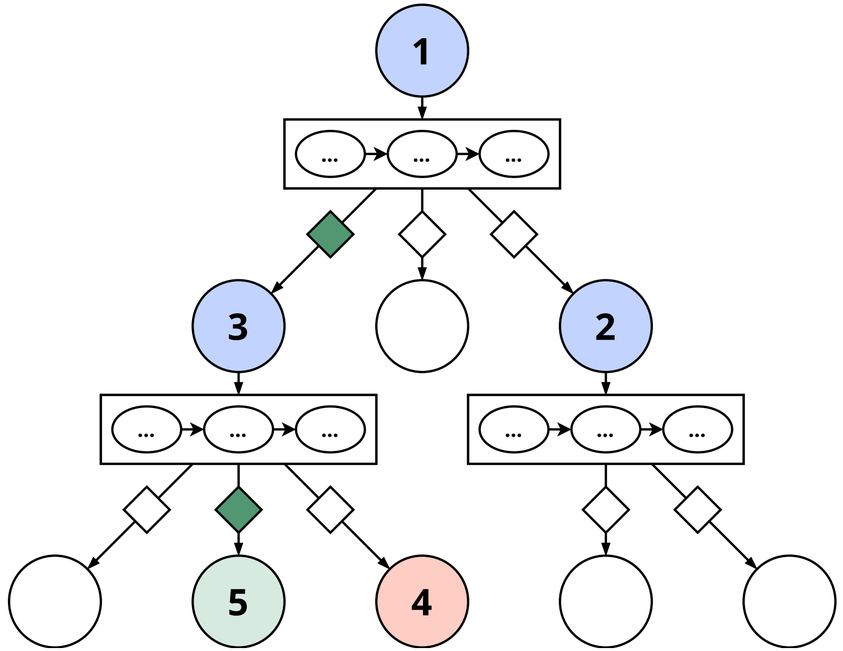


They could also **complete partial proofs** or **fill in missing steps**

Discussion: Reasoning Models

What about **reasoning** models?



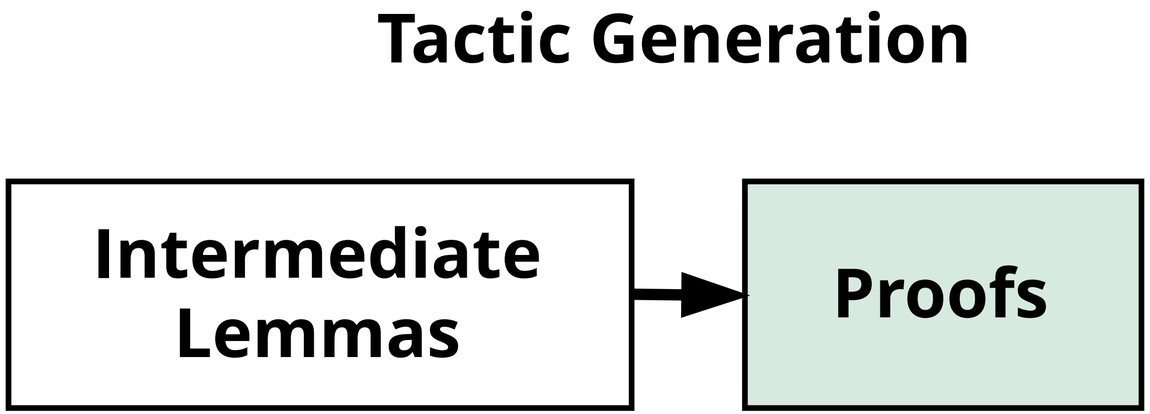


❌ No interaction with ITP

⏱ High resource consumption

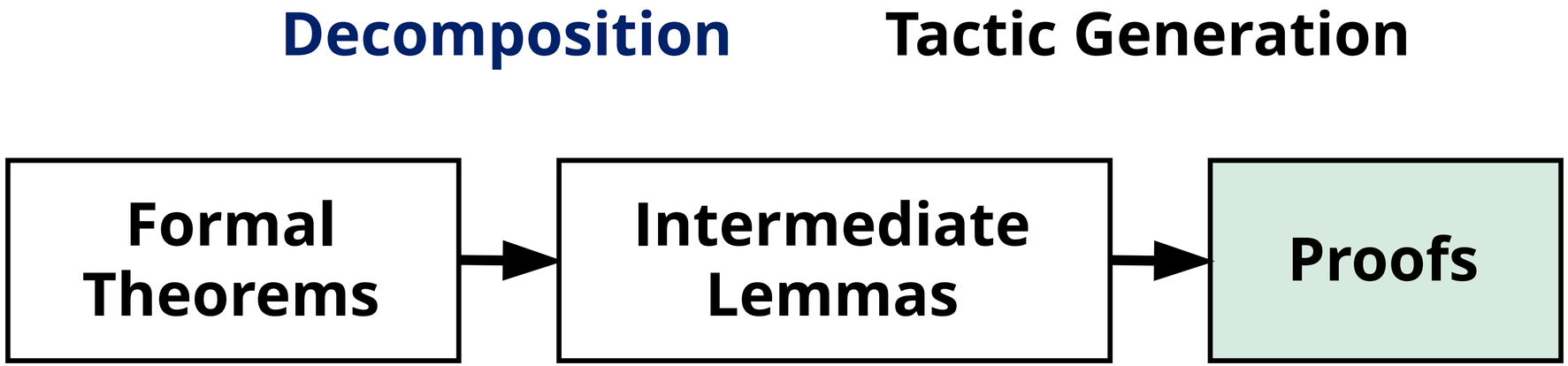
Discussion: Proof Decomposition

Can LLMs generate **intermediate lemmas**?



Discussion: Proof Decomposition

Can LLMs generate **intermediate lemmas**?



Conclusion

Can LLMs **augment or replace** the manual verification of system software?

- LLMs can prove a surprising fraction of FSCQ
- Hints significantly improve proof completion rate

Further Discussion / Open Questions

Human Augmentation?

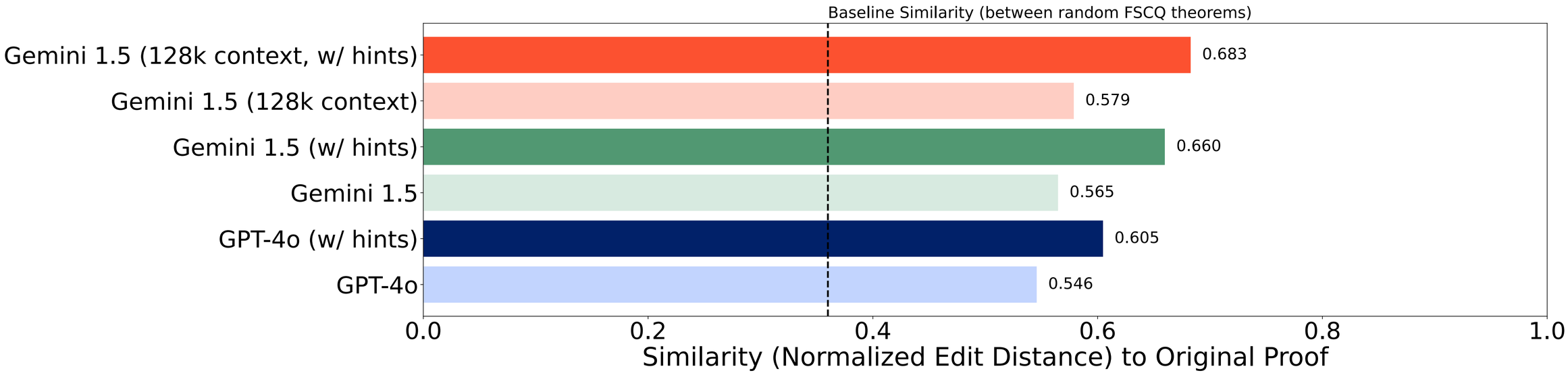
Proof Decomposition?

Reasoning Models?

Context Retrieval?

Open Source (Code and Generated Proofs)

Evaluation: Are models memorizing FSCQ's proofs?



Generated proofs are **not direct copies** of the originals