Where Does Proof Search Spend Its Effort?

A Visualization and Profiling Tool for Formal NLI

Koharu Saeki　Daisuke Bekki

Ochanomizu University

NALOMA VI　ESSLLI 2026, Prague　August 6th, 2026

Proof-Theoretic Semantics for NLI

We approach NLI from the perspective of **proof-theoretic semantics**:

The entailment relation is characterized as **the existence of a proof** of the semantic representation of the hypothesis from those of the premises.

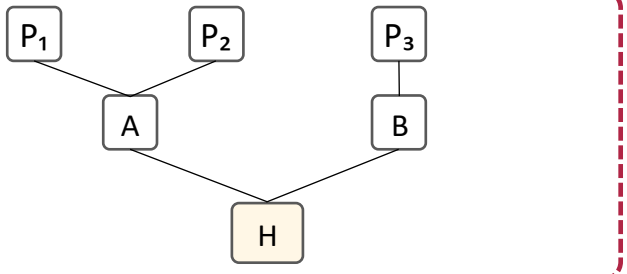
When entailment holds:

Entailment judgment

Entailment
(holds)

+

Constructed proof (proof diagram)

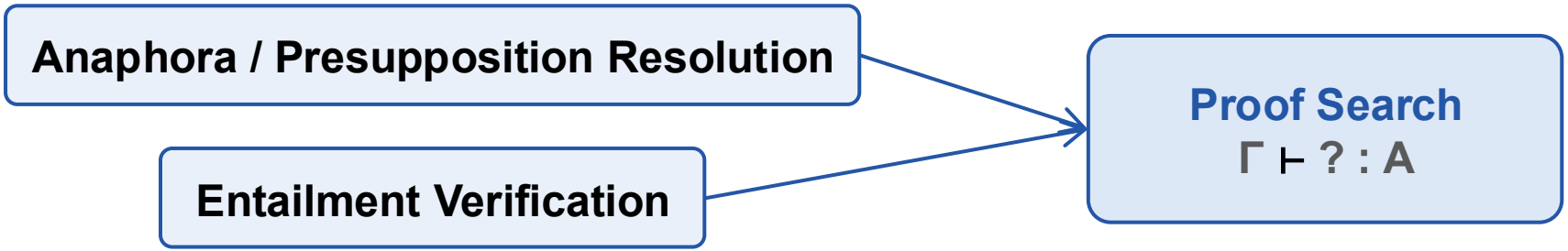


Dependent Type Semantics (DTS; Bekki, 2014; Bekki and Mineshima, 2017)

- A semantic framework based on Dependent Type Theory (DTT; Martin-Löf, 1984)

A semantic representation is a **type** of DTT

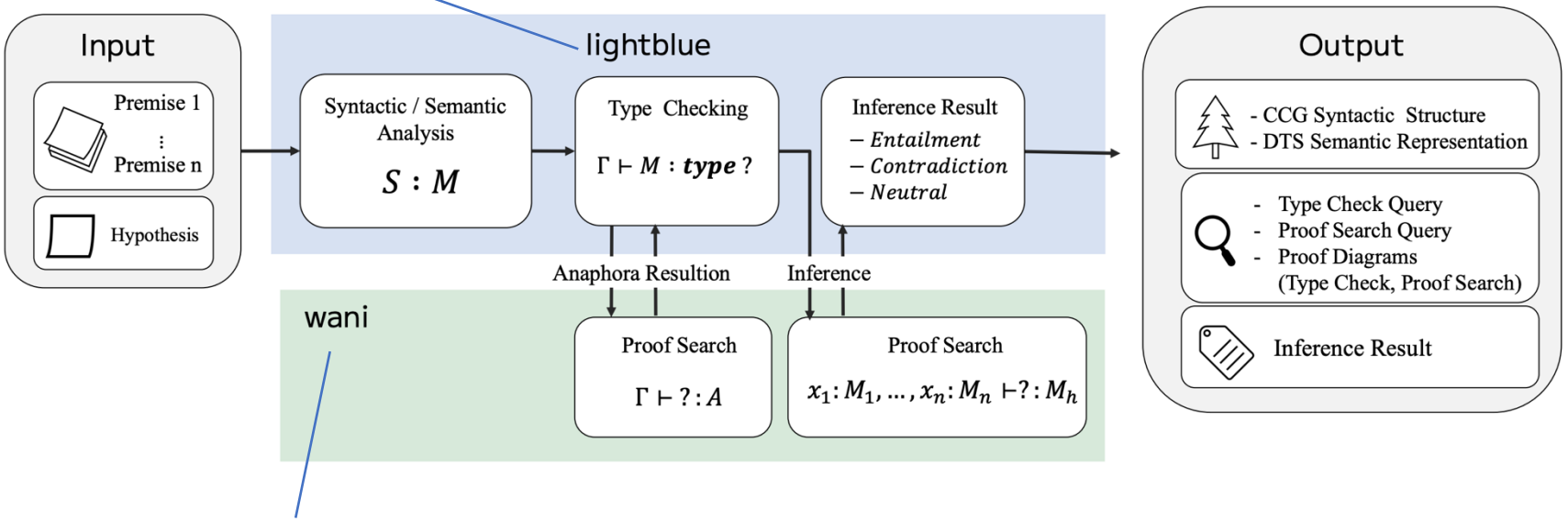
→ proving it means **constructing a term of that type**



Two tasks, **one operation**: proof search.

Overview of the inference pipeline of lightblue

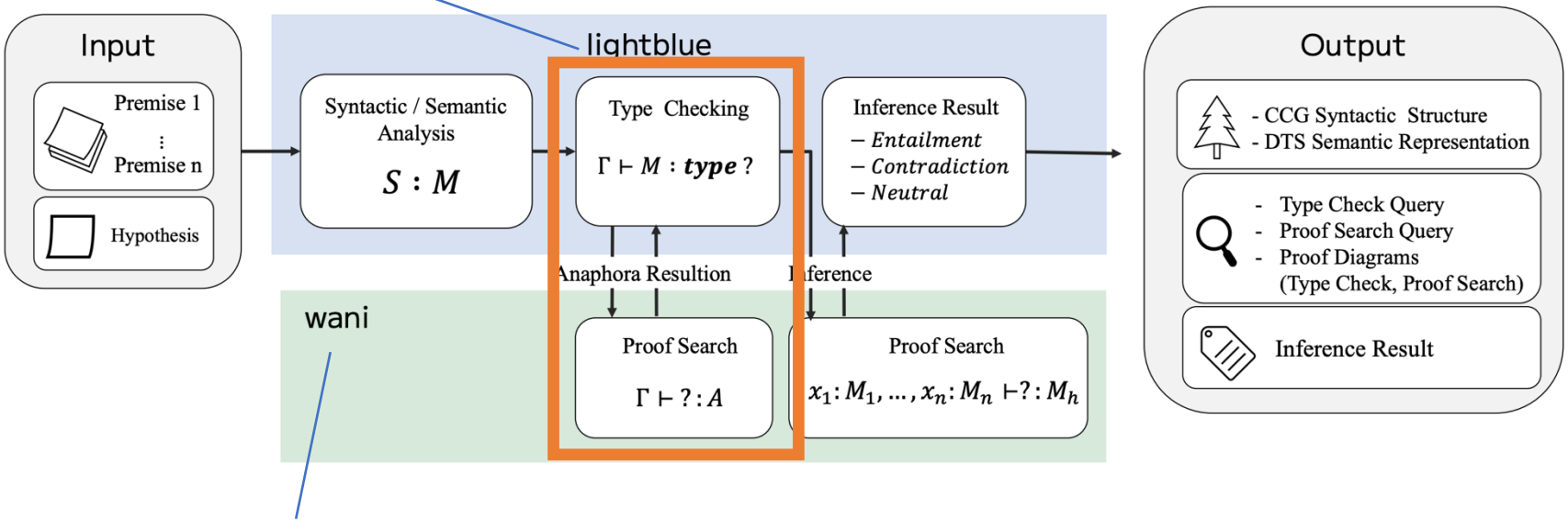
lightblue (Tomita et al., 2025): logic-based NLI system



wani (Daido and Bekki, 2020): an automated theorem prover for DTT

Proof Search in Type Checking

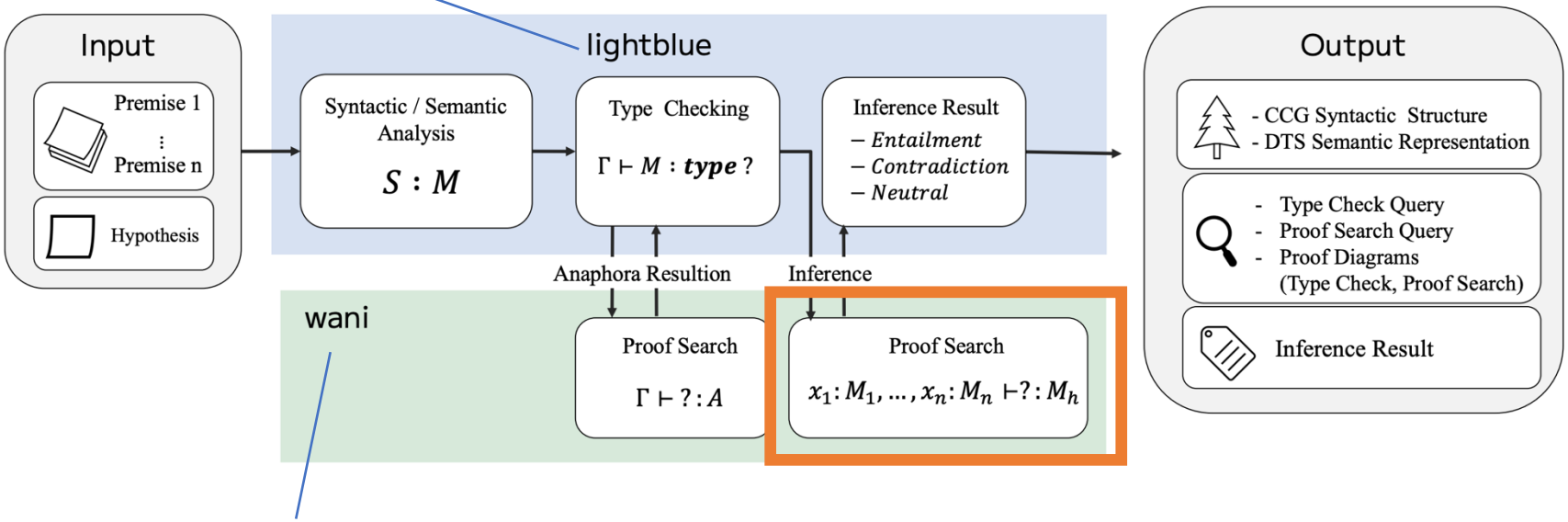
lightblue (Tomita et al., 2025): logic-based NLI system



wani (Daido and Bekki, 2020): an automated theorem prover for DTT

Proof Search in Entailment Judgment

lightblue (Tomita et al., 2025): logic-based NLI system

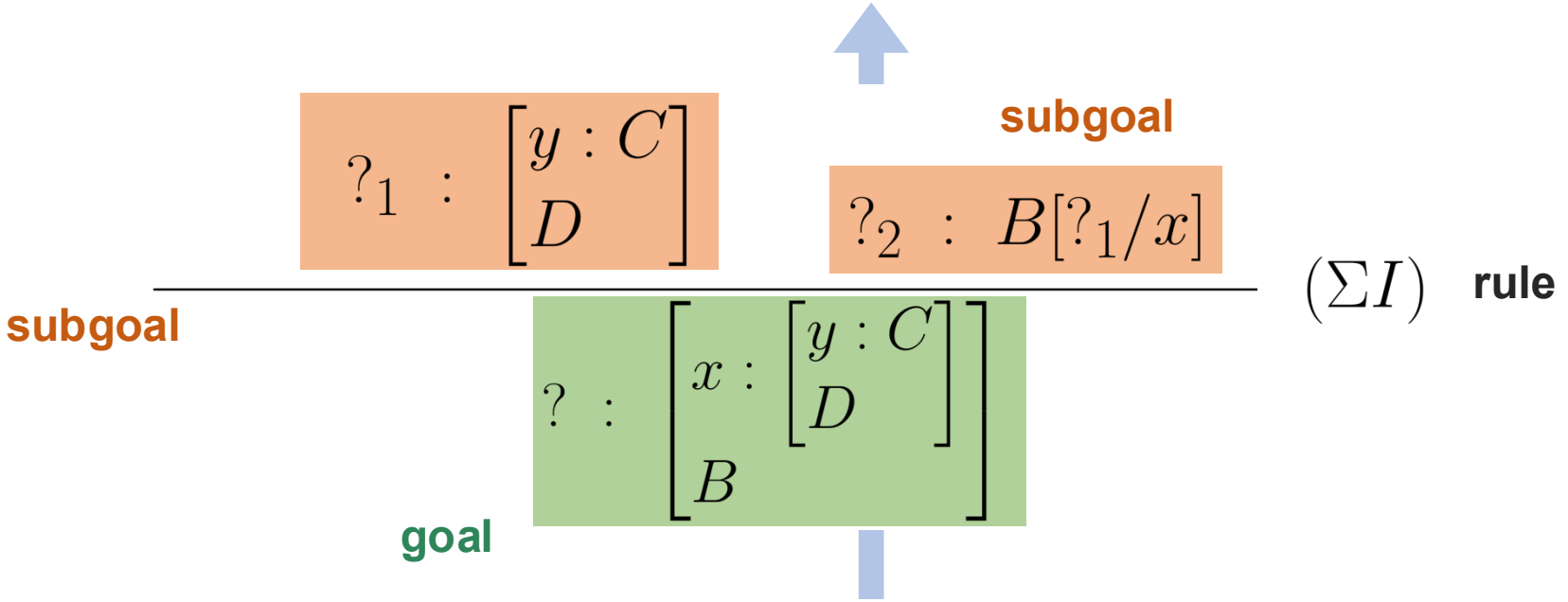


wani (Daido and Bekki, 2020): an automated theorem prover for DTT

How *wani* Searches: Backward Reasoning

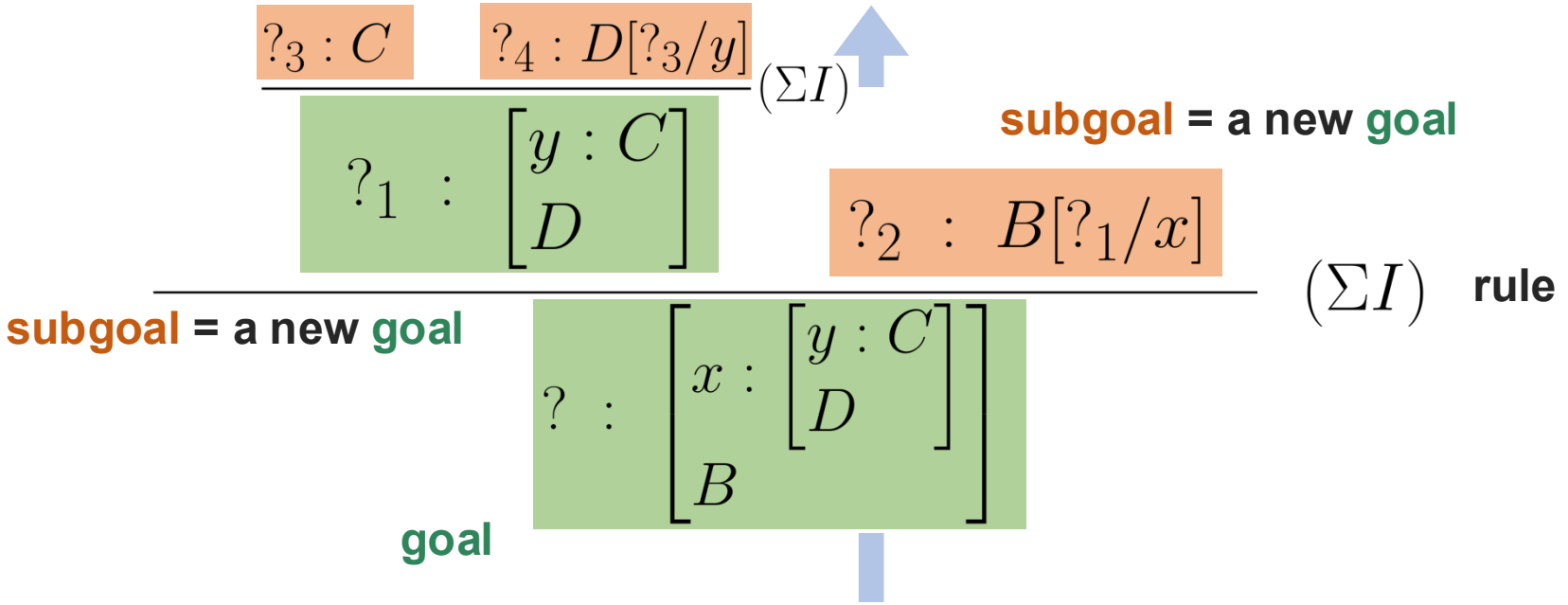
- *wani* uses **backward reasoning** as its basic search strategy

asks what is needed to prove the goal



How *wani* Searches: Backward Reasoning

- Applying a rule decomposes **a goal** into **subgoals**. Each **subgoal** becomes a new **goal**, and the same procedure is applied **recursively**



How *wani* Searches: Forward Reasoning

- Some rules use **forward reasoning**

derives what follows from the premises

forward reasoning

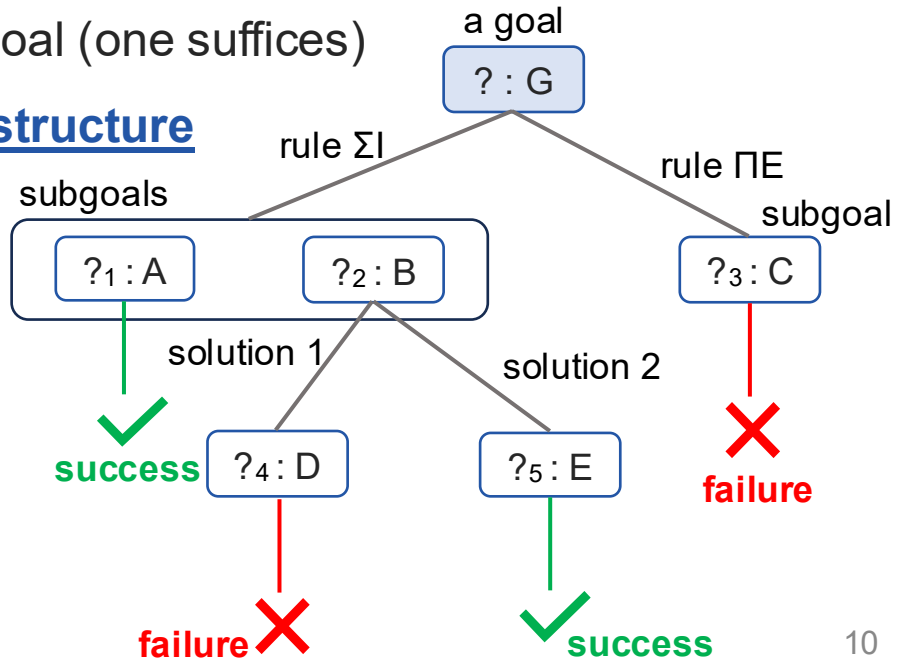
$$\frac{s \;:\; \begin{bmatrix} x : A \\ B \end{bmatrix}}{\pi_1(s) \;:\; A}\,(\Sigma E)$$

$$\frac{?_1 : A \qquad ?_2 : B[?_1/x]}{? \;:\; \begin{bmatrix} x : A \\ B \end{bmatrix}}\,(\Sigma I)$$

How *wani* Searches: Branching into a Tree

- one rule application can produce **multiple subgoals** (all must be discharged)
- multiple rules can derive a goal (wani automatically tries **up to 14 inference rules** for each goal)
- multiple solutions can discharge a subgoal (one suffices)

→ the search **branches** and forms **a tree structure**



How *wani* Controls The Search

- Proof search in DTT is undecidable in general

→ Search is executed as **iterative-deepening depth-first search**, bounded by:

Depth Limit	**Time Limit**	**Loop Avoidance**
Do not explore paths deeper than the limit	Stop the search when the time limit is reached	Avoid revisiting the same failed goals

Where Does the Search Spend Its Effort?

- The search can grow large: on a single JSeM (Kawazoe et al., 2015) problem, **hundreds of goals** are generated, and **most of them fail**(Failed exploration is the intrinsic cost of proof search).

→ Improvement starts from **knowing where that effort goes.**

We want to know four things:

Structure
which branches were explored, and how deep did they go?

Time
where is the time spent?

Rule outcomes
which rules are tried, and which succeed?

Failures
which failures repeat?

How Is Search Recorded?

The Only Record: Textual Debug

```
0 current goal : G |- ? : (S entity)(S cry(hanako,0))(S tense(1))T
0 ... is not acceptable type in PiF
0 ... is not acceptable type in SigmaF
0-acceptable [SubGoalSet SigmaI ...]   0-acceptable [SubGoalSet EFQ ...]
  with SigmaI, want to prove [SubGoal entity, SubGoal S cry. S tense]
  1 current goal : G |- ? : entity
  1 depth @ deduce : entity
0 deduce failed
```

- Inserted by developers to inspect **the behavior at specific points** during debugging
- Record individual goals, rule applications, search outcomes (success/failure, depth, etc.), …

Problem: Lack of Analyzability

- Existing logs are not sufficient for **analyzing the search as a whole.**

Structure
A goal's parent and branch are not linked: the tree cannot be rebuilt reliably.

Time
No timestamps: where the time goes cannot be recovered.

Rule outcomes
Starts and outcomes are unlinked: no attribution to a goal or rule.

Failures
No goal IDs: repetitions of the same goal cannot be told apart.

Our goal: Make the search process analyzable.

Proposal: A Profiling and Visualization Tool

Its design follows one principle: **record** everything, **analyze** afterwards.

Step 1: Record (inside *wani*)

One event per search step

Fields: kind, timestamp, goal ID, rule name, subgoal index

→ **a flat event log**

Step 2: Analyze (outside *wani*)

Rebuild the search tree

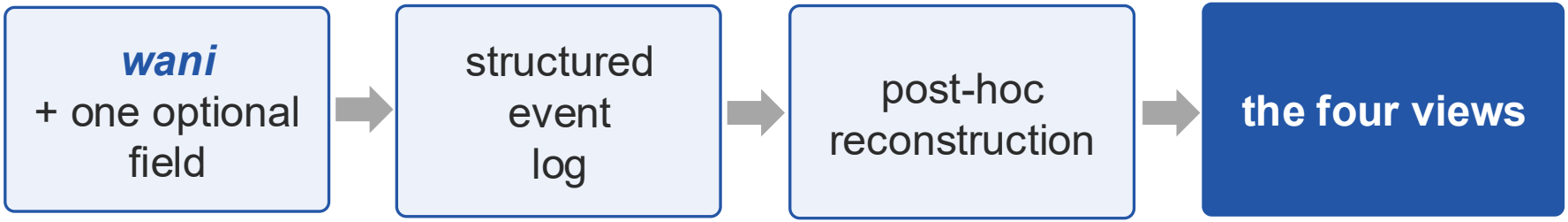
Aggregate outcomes per rule

Classify repeated failures

→ **the four views**

The prover is barely touched: one optional field

Architecture: One Field to Four Views



Nothing downstream is wani-specific: emit the same events from another prover and the dashboard is reused unchanged (e.g., ccg2lambda; Mineshima et al., 2015).

Running Example: JSeM 720

JSeM (Kawazoe et al., 2015) is a curated Japanese NLI dataset.

Premise		Hypothesis
Taro-ga Hanako-ni naka-re-ta.	$\models$	Hanako-ga nai-ta.
`Taro-NOM Hanako-DAT cry-PASS-PST`		`Hanako-NOM cry-PST`
'Taro was affected by Hanako's crying.'		*'Hanako cried.'*

Running Example: JSeM 720 and Its Proof Search Query

lightblue converts the input sentences into **a proof search query, the initial goal**:

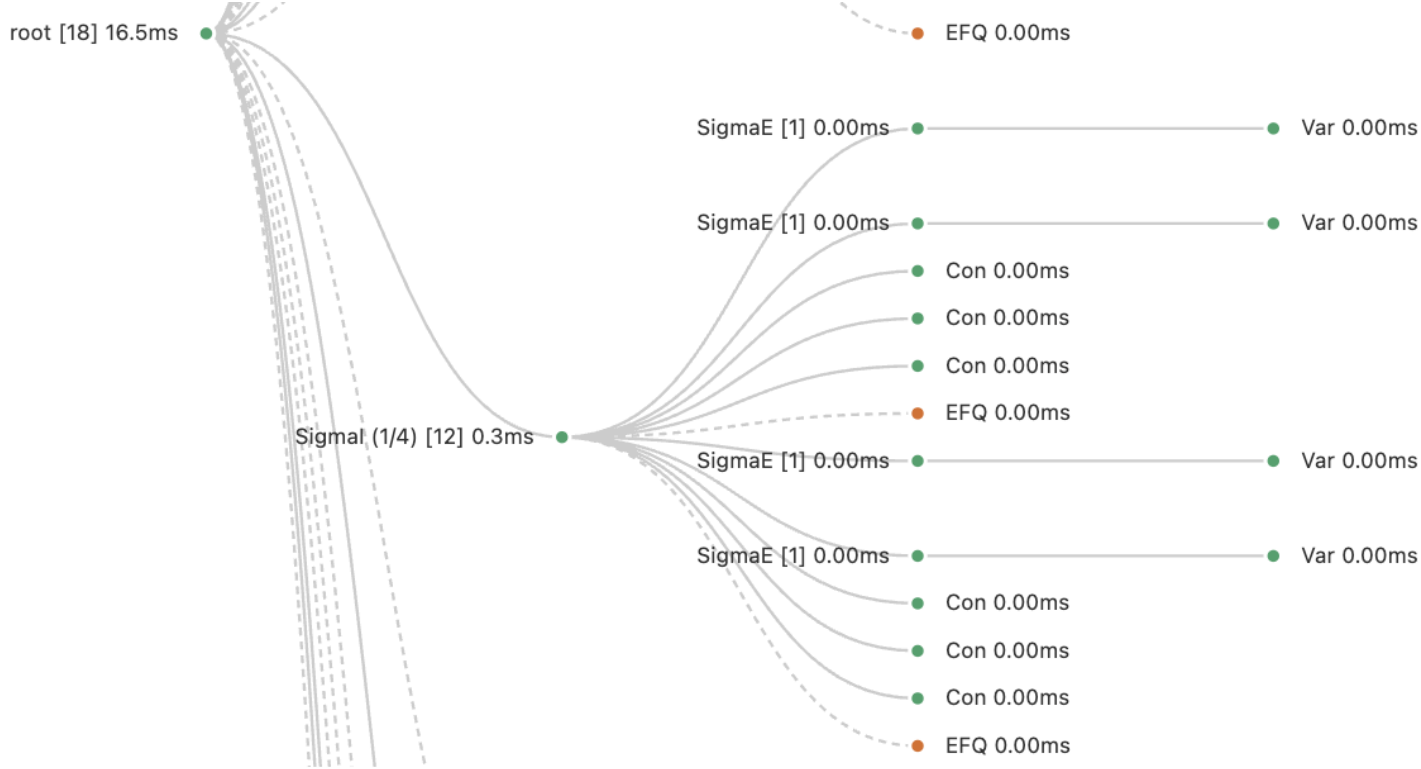
the semantic representations of the premises

$$s_0: \left[\left[\left[\begin{array}{l} x_0: \text{entity} \\ \left[\begin{array}{l} u_0: \text{泣く/なく/ガ}(x_0, \text{花子/はなこ}) \\ \left[\left[\begin{array}{l} x_1: \text{entity} \\ \left[\begin{array}{l} u_1: \#\text{迷惑}(x_1, \text{太郎/たろう;太郎/たろう}, x_0) \\ \#\text{タ}(x_0) \end{array}\right] \end{array}\right]\right] \end{array}\right] \end{array}\right]\right]\right] \vdash ?: \left[\left[\begin{array}{l} x_2: \text{entity} \\ \left[\begin{array}{l} u_3: \text{泣く/なく/ガ}(x_2, \text{花子/はなこ}) \\ \#\text{タ}(x_2) \end{array}\right] \end{array}\right]\right]$$

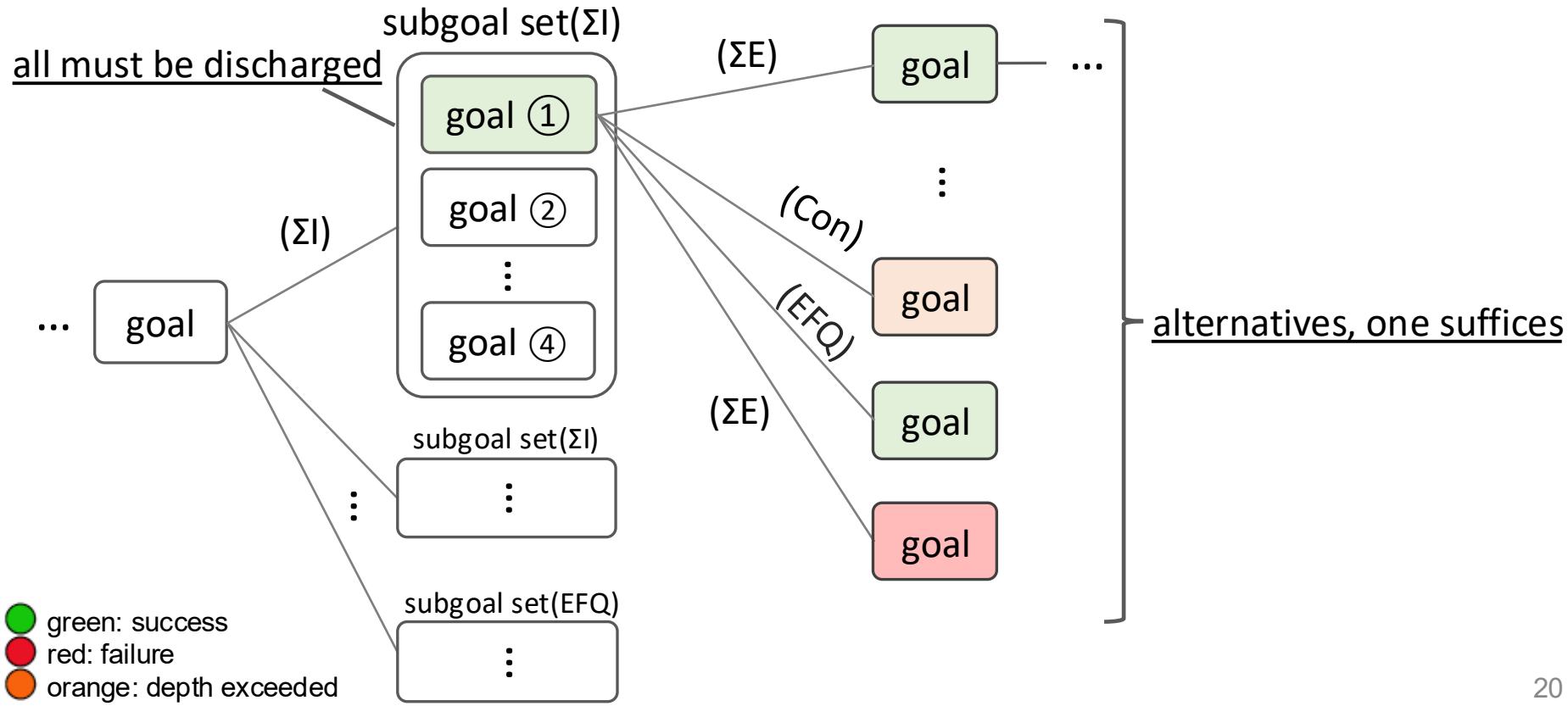
the semantic representation of the hypothesis

***wani* proves this entailment**: under a certain search setting, it finds two proof terms.

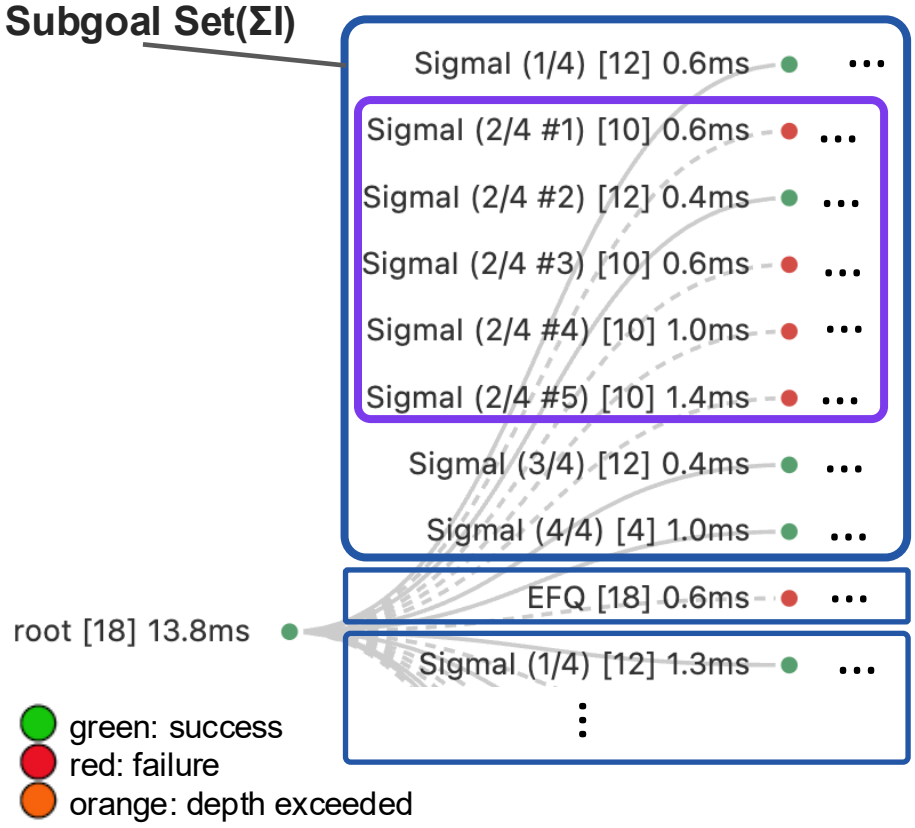
Search Tree: What Was Explored, and How Deep?



Search Tree: What Was Explored, and How Deep?



Search Tree: What Was Explored, and How Deep?



How to read a node label

branch 2 of five : **OR** (one branch is enough)

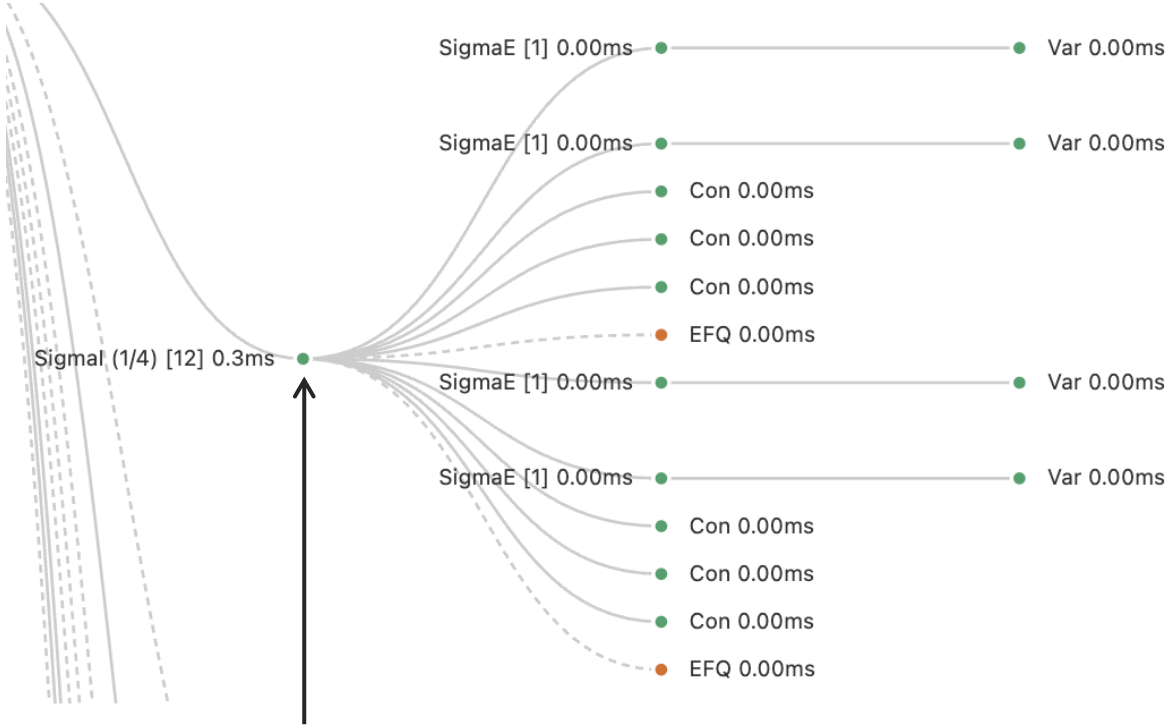
a rule — time in this subtree

Sigmal (2/4 #2) [12] 0.4ms

children explored

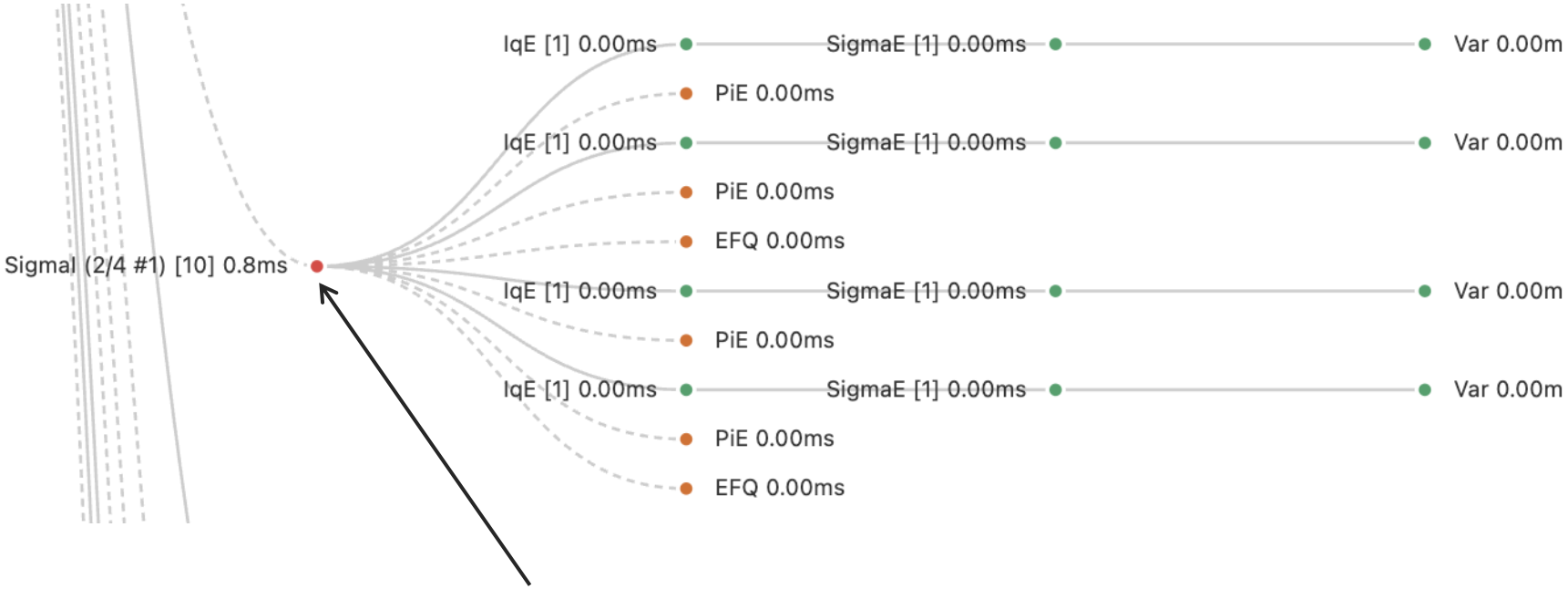
subgoal 2 of 4 : **AND** (all four must be discharged)

Search Tree: What Was Explored, and How Deep?



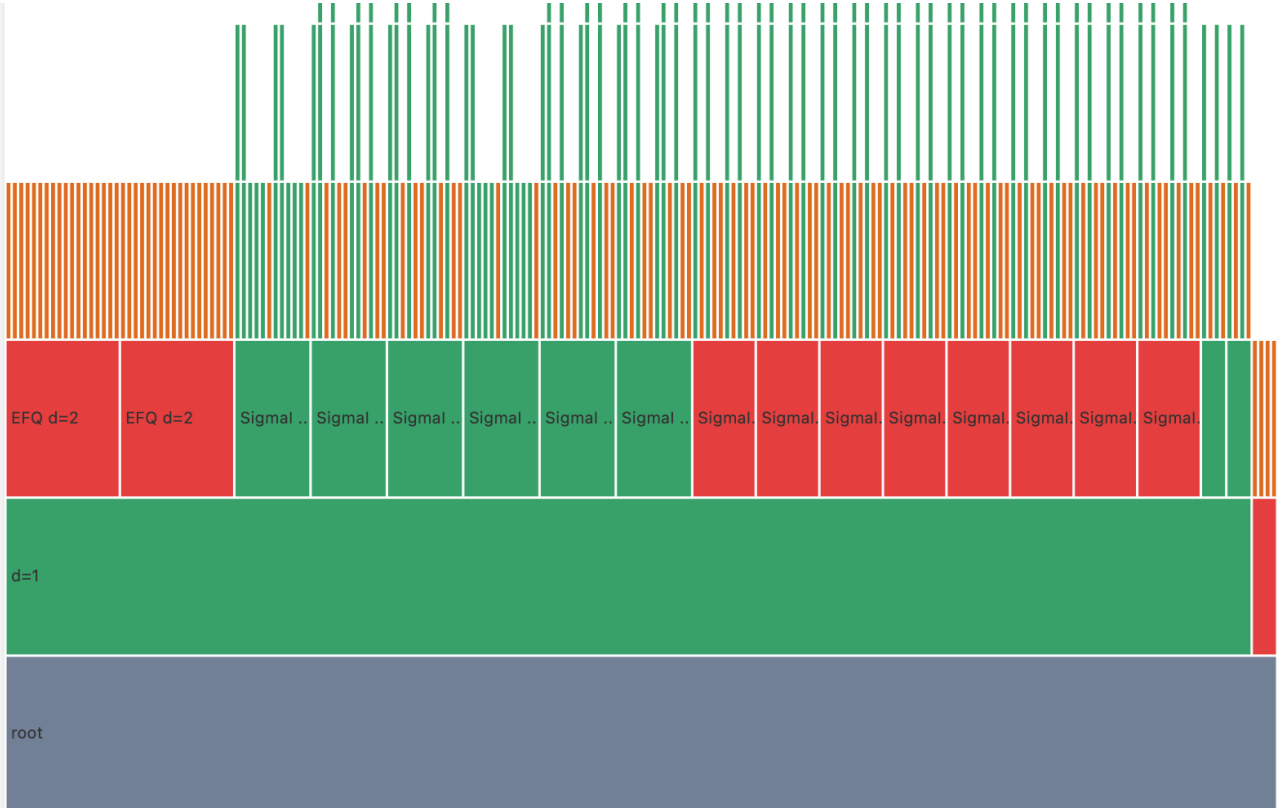
ΣI subgoal 1 of 4: succeeds; leaves close by forward reasoning

Search Tree: What Was Explored, and How Deep?

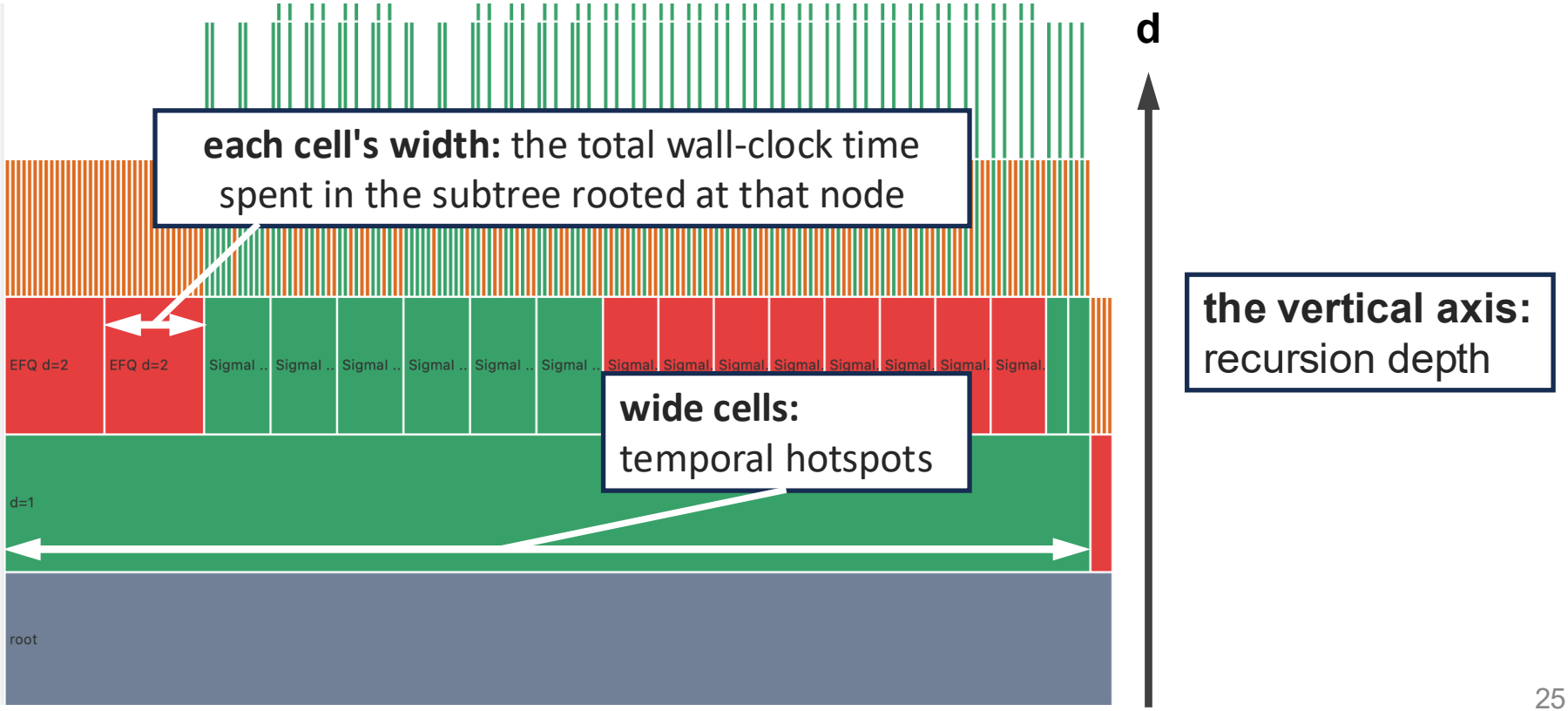


ΣI subgoal 2, branch #1: fails; its attempts hit the depth bound

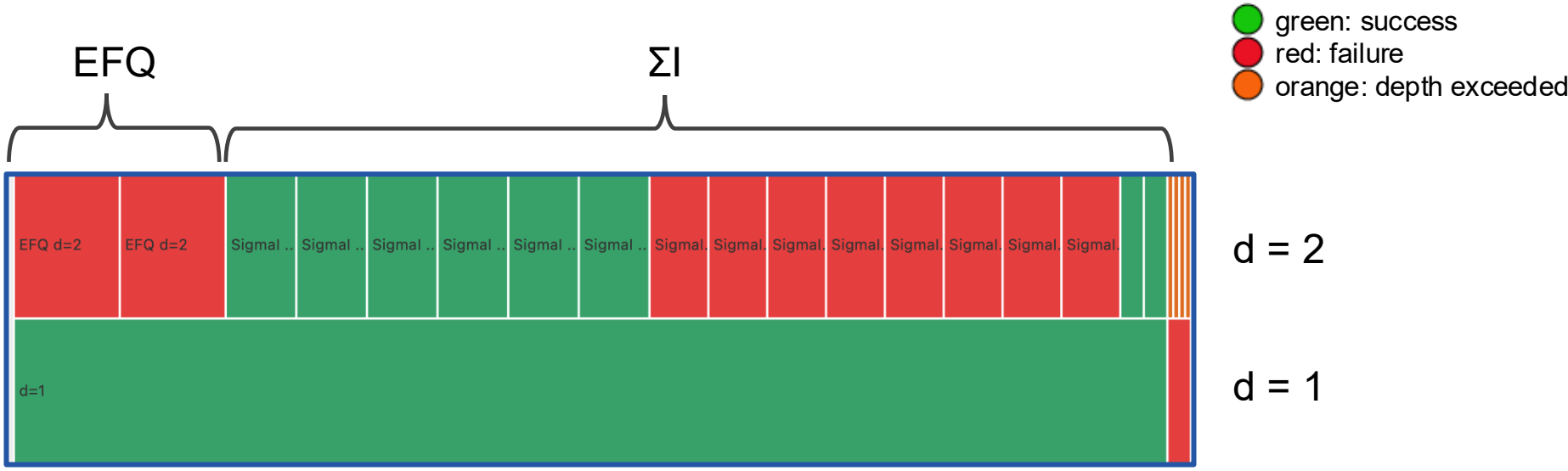
Flame Graph: Where Is the Time Spent?(1/3)



Flame Graph: Where Is the Time Spent?(2/3)



Flame Graph: Where Is the Time Spent?(3/3)



Answer: ΣI dominates roughly 90% of wall-clock time.

Rule Statistics: Which Rules Tried, and Which Succeed?

Rule Statistics

Rule	Attempts	Success	Fail	Depth	Loop	Time	Pending	Rate	Total (ms)
Con	12	12	0	0	0	0	0	100.0%	0.01
EFQ	40	0	2	38	0	0	0	0.0%	1.3
IqE	48	48	0	0	0	0	0	100.0%	0.1
PiE	48	0	0	48	0	0	0	0.0%	0.03
SigmaE	68	68	0	0	0	0	0	100.0%	0.10
SigmaF	32	0	0	32	0	0	0	0.0%	0.03
SigmaI	18	8	8	2	0	0	0	44.4%	11.5
Var	68	68	0	0	0	0	0	100.0%	0.04

Rule Statistics: Which Rules Tried, and Which Succeed?

rule applied, subgoals generated

subtree discharged the goal

Rule Statistics

Rule	Attempts	Success	Fail	Depth	Loop	Time	Pending	Rate	Total (ms)
Con	12	12	0	0	0	0	0	100.0%	0.01
EFQ	40	0	2	38	0	0	0	0.0%	1.3
IqE	48	48	0	0	0	0	0	100.0%	0.1
PiE	48	0	0	48	0	0	0	0.0%	0.03
SigmaE	68	68	0	0	0	0	0	100.0%	0.10
SigmaF	32	0	0	32	0	0	0	0.0%	0.03
SigmaI	18	8	8	2	0	0	0	44.4%	11.5
Var	68	68	0	0	0	0	0	100.0%	0.04

Rule Statistics: Which Rules Tried, and Which Succeed?

Rule Statistics

Rule	Attempts	Success	Fail	Depth	Loop	Time	Pending	Rate	Total (ms)
Con	12	12	0	0	0	0	0	100.0%	0.01
EFQ	40	0	2	38	0	0	0	0.0%	1.3
IqE	48	48	0	0	0	0	0	100.0%	0.1
PiE	48	0	0	48	0	0	0	0.0%	0.03
SigmaE	68	68	0	0	0	0	0	100.0%	0.10
SigmaF	32	0	0	32	0	0	0	0.0%	0.03
SigmaI	18	8	8	2	0	0	0	44.4%	11.5
Var	68	68	0	0	0	0	0	100.0%	0.04

Con, =E, ΣE, Var: succeed every time and cost almost nothing.

Rule Statistics: Which Rules Tried, and Which Succeed?

Rule Statistics

Rule	Attempts	Success	Fail	Depth	Loop	Time	Pending	Rate	Total (ms)
Con	12	12	0	0	0	0	0	100.0%	0.01
EFQ	40	0	2	38	0	0	0	0.0%	1.3
IqE	48	48	0	0	0	0	0	100.0%	0.1
PiE	48	0	0	48	0	0	0	0.0%	0.03
SigmaE	68	68	0	0	0	0	0	100.0%	0.10
SigmaF	32	0	0	32	0	0	0	0.0%	0.03
SigmaI	18	8	8	2	0	0	0	44.4%	11.5
Var	68	68	0	0	0	0	0	100.0%	0.04

EFQ, πE, ΣF: Zero Successes

Failure Analysis: Which Failures Repeat?

Deduce Failed (6)

2 d=2 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- (Σ entity)(Σ ((泣く/なく/ガ ...

2 d=2 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ?:((泣く/なく/ガ 花子/はなこ) ...

2 d=2 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ?:((泣く/なく/ガ 花子/はなこ) ...

2 d=2 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ?:((泣く/なく/ガ 花子/はなこ) ...

2 d=2 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ?:((泣く/なく/ガ 花子/はなこ) ...

1 d=1 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ?:(Σ entity)(Σ ((泣く/なく/ガ ...

Depth Exceeded (15)

40 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- 花子/はなこ:entity

32 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T, entity, ((泣く/なく/ガ 花子/は...

4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ((泣く/なく/ガ 花子/はなこ) d...

4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ((泣く/なく/ガ 花子/はなこ) π...

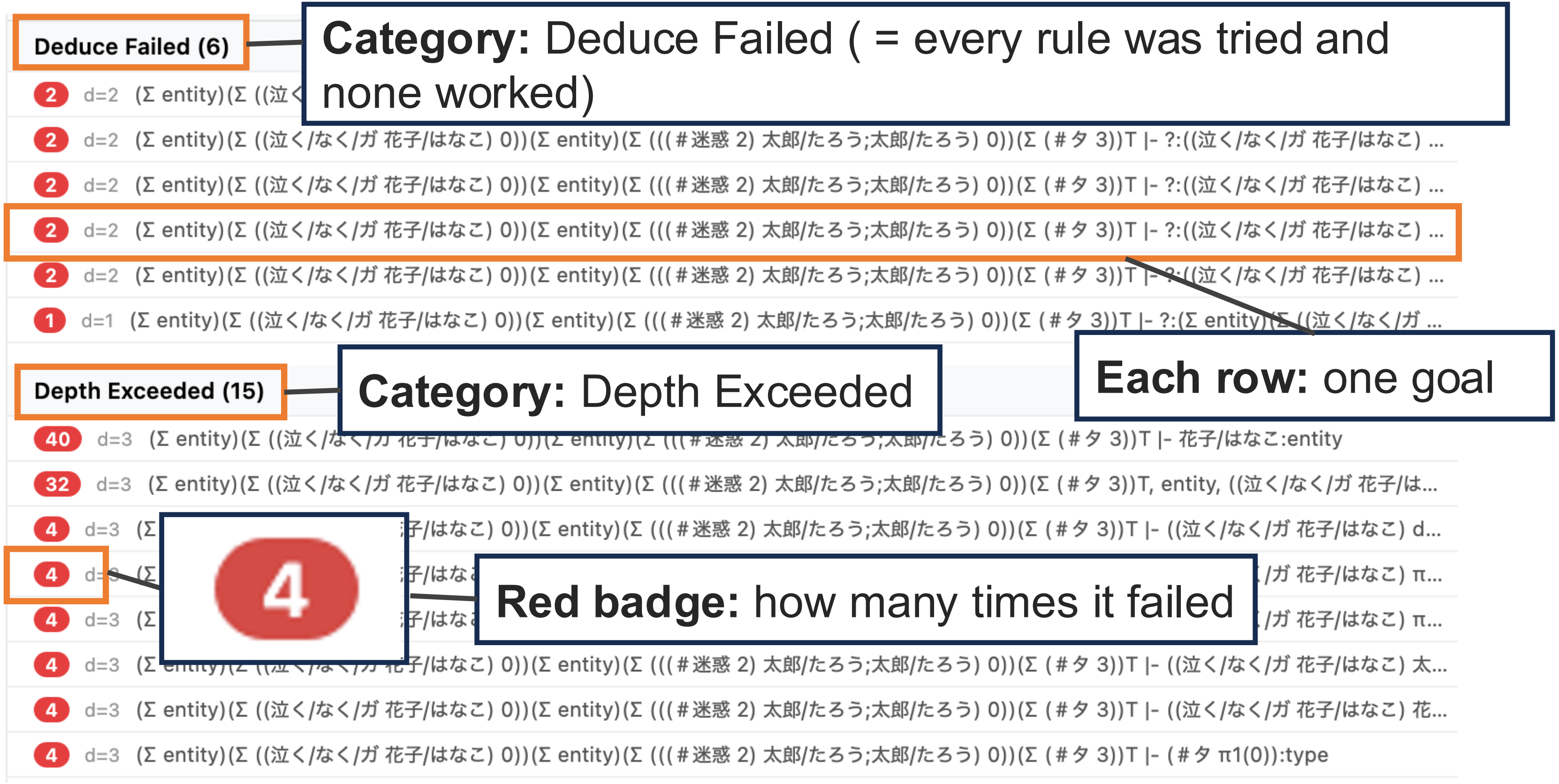
4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ((泣く/なく/ガ 花子/はなこ) π...

4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ((泣く/なく/ガ 花子/はなこ) 太...

4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- ((泣く/なく/ガ 花子/はなこ) 花...

4 d=3 (Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((#迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (#夕 3))T |- (#夕 π1(0)):type

Failure Analysis: Which Failures Repeat?



Failure Analysis: Which Failures Repeat?

Depth Exceeded, top rows:

Depth Exceeded (15)

40	d=3	(Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((＃迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (＃タ 3))T |- 花子/はなこ:entity
32	d=3	(Σ entity)(Σ ((泣く/なく/ガ 花子/はなこ) 0))(Σ entity)(Σ (((＃迷惑 2) 太郎/たろう;太郎/たろう) 0))(Σ (＃タ 3))T, entity, ((泣く/なく/ガ 花子/は…

Answer: the same goals fail again and again: one goal fails 40 times.

What This Gives ML-Based Acceleration (1/2)

- Learned proof search is advancing in ITPs (LeanDojo, Yang et al., 2023; Graph2Tac, Blaauwbroek et al., 2024). **Neural Wani** (Miyagawa et al., 2026) brings it to *wani*.

What This Gives ML-Based Acceleration (2/2)

1. **Pruning candidates, measured**
   on JSeM 720, ΠE, ΣF, EFQ: many attempts, zero contribution → the kind of per-query signal a learned rule ranker could use

2. **Failure paths can become training data**
   previously unusable as data; the event log makes them recoverable → positive and negative examples for learning

3. **Beyond hand-optimization**
   wani already prunes by hand and most exploration still fails → the measured case that learning is needed

Structural Faithfulness

The tree is rebuilt post-hoc, with no parent pointers.
An error would silently distort every view, so we verify three invariants.

goal pairing: every GoalStart has exactly one GoalEnd

outcome uniqueness: at most one terminal outcome per goal

depth invariant: each goal at depth d > 1 has an active predecessor at d−1

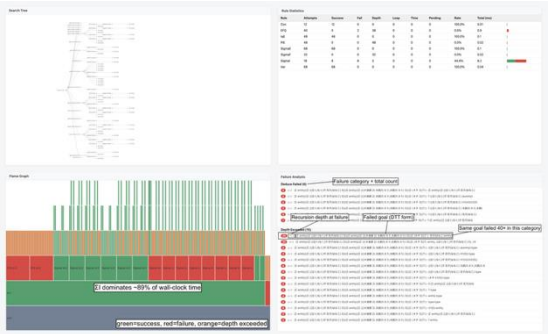
Check	JSeM 720 / 336 goals	JSeM 699 / 1,680 goals
all three invariants	**pass**	**pass**

Conclusion

The Search Process Is Now Analyzable

- Profiling & visualization tool for proof search in DTT-based NLI provers
- Case study on JSeM surfaces concrete bottlenecks invisible in raw logs
- Structured event log can become training data for ML acceleration

Future Work: corpus-level aggregation over JSeM, porting via the shared event schema, training Neural Wani on the collected signals



Try it now

Live demo + Docker: wani-viz-site.fly.dev

Acknowledgements

This work was supported in part by JST CREST Grant Number JPMJCR2565 and JSPS KAKENHI Grant Number JP23H03452, Japan.

References

- Daisuke Bekki. 2014. Representing anaphora with dependent types. In Logical Aspects of Computational Linguistics (LACL 2014), LNCS 8535, pages 14–29. Springer.
- Daisuke Bekki and Koji Mineshima. 2017. Context-Passing and Underspecification in Dependent Type Semantics, pages 11–41. Springer International Publishing, Cham.
- Hinari Daido and Daisuke Bekki. 2020. Development of an automated theorem prover for the fragment of DTS. In Proceedings of the 17th International Workshop on Logic and Engineering of Natural Language Semantics (LENLS17).
- Lasse Blaauwbroek, Miroslav Olšák, Jason Rute, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. 2024. Graph2tac: Online representation learning of formal math concepts. In International Conference on Machine Learning.
- Brendan Gregg. 2016. The flame graph. Commun. ACM, 59(6):48–57.
- Ai Kawazoe, Ribeka Tanaka, Koji Mineshima, and Daisuke Bekki. 2015. An inference problem set for Evaluating Semantic Theories and Semantic Processing Systems for Japanese. In Proceedings of the Twelfth International Workshop on Logic and Engineering of Natural Language Semantics (LENLS12), pages 67–73, Tokyo-Yokohama, Japan.
- Per Martin-Löf. 1984. Intuitionistic Type Theory Vol. 1. Bibliopolis.
- Pascual Martínez-Gómez, Koji Mineshima, Yusuke Miyao, and Daisuke Bekki. 2016. ccg2lambda: A compositional semantics system. In Proceedings of ACL-2016 System Demonstrations, pages 85–90, Berlin, Germany. Association for Computational Linguistics.
- Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2055–2061, Lisbon, Portugal. Association for Computational Linguistics.
- Nanako Miyagawa, Hinari Daido, and Daisuke Bekki. 2026. Neural Wani: Izon gata riron no tame no jidou teiri shoumeiki wani no kousokuka ni mukete (trans. neural Wani: Toward speeding up the automated theorem prover Wani for dependent type theory). In Proceedings of the 32nd Annual Meeting of the Association for Natural Language Processing, pages 2930–2934. In Japanese.
- Leonardo de Moura and Sebastian Ullrich. 2021. The lean 4 theorem prover and programming language. In Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings, page 625–635, Berlin, Heidelberg. Springer-Verlag.
- Alex Sanchez-Stern, Yousef Alhessi, Lawrence Saul, and Sorin Lerner. 2020. Generating correctness proofs with neural networks. In Proceedings of the 4th ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, MAPL 2020, page 1–10, New York, NY, USA. Association for Computing Machinery.
- Peiyang Song, Kaiyu Yang, and Anima Anandkumar. 2025. Lean copilot: Large language models as copilots for theorem proving in lean. Preprint, arXiv:2404.12534.
- Mark Steedman. 2000. The Syntactic Process. MIT Press, Cambridge, MA.
- Amitayush Thakur, George D. Tsoukalas, Yeming Wen, Jimmy Xin, and Swarat Chaudhuri. 2023. An In-Context Learning Agent for Formal Theorem-Proving.
- The Coq Development Team. 2024. The Coq Proof Assistant Reference Manual.
- Asa Tomita, Mai Matsubara, Hinari Daido, and Daisuke Bekki. 2025. Natural language inference with CCG parser and automated theorem prover for DTS. In Proceedings of the Second Workshop on the Bridges and Gaps between Formal and Computational Linguistics (BriGap-2), pages 1–7, Düsseldorf, Germany. Association for Computational Linguistics.
- Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. In Advances in Neural Information Processing Systems, volume 36, pages 21573–21612. Curran Associates, Inc.