Towards a Universal Interactive Theorem Proving Interface

Kaustuv Chaudhuri

Inria & IP Paris, France

Reykjavík, Iceland
2025-09-29

Obstacle 1: No Universal Language

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda, Abella

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda, Abella, …

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda, Abella, …

- Not many people are proficient in multiple systems

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda, Abella, …

- Not many people are proficient in multiple systems
- Different logics, proof styles, tactics, solvers, …

Obstacle 1: No Universal Language

Poll: Raise your hand if you have used

Rocq or Lean, Isabelle{/HOL} or HOL{4, -Light}, ACL2, Mizar, Agda, Abella, …

- Not many people are proficient in multiple systems
- Different logics, proof styles, tactics, solvers, …
- Hard to do interactivity in a system agnostic way

Obstacle 1: No Universal File System

Poll: Raise your hand if you have used

NTFS (Windows), apfs (macOS), ext4 (Linux), NFS, Dropbox, Google Drive, IPFS, …

- *Not many people are proficient in multiple file systems*
- *Different system calls, libraries, naming conventions, …*
- *Hard to do interactivity in a system agnostic way?*

Obstacle 2: Human Computer Interfaces

Obstacle 2: Human Computer Interfaces

Obstacle 2: Human Computer Interfaces

Obstacle 2: Human Computer Interfaces

Obstacle 2: Human Computer Interfaces





Obstacle 2: Some Human Computer Interfaces





Obstacle 2: Some Human Computer Interfaces





- Traditional computer: {  ,  }
- Mobile computer: {  ,  }

Obstacle 3: The Irresistible Allure of Black Boxes

Obstacle 3: The Irresistible Allure of Black Boxes

Claude Can (Sometimes) Prove It

By: Mike Dodds September 16, 2025

Let me get right to the point without any nonsense about aliens:

1. Claude Code, the new AI coding agent from Anthropic, is pretty good at interactive theorem proving (ITP).
2. I find this very surprising, and you probably should too.

Interactive theorem proving tools such as Lean are the most powerful and trustworthy kind of formal methods tool. They have been used to formally verify important things such as cryptographic libraries, compilers, and operating systems. Unfortunately, even experts find ITP proofs time-consuming and error-prone. That's why it's exciting—and very surprising!—to find that Claude Code is so good at ITP. Today, Claude Code can complete many complex proof steps independently, but it still needs a 'project manager' (me) to guide it through the whole formalization. But I think Claude Code points to a world where experts aren't necessary, and theorem provers can be used by many more people.

The rest of this post digs into what Claude Code can actually do. But if you're interested in automated reasoning or formal verification, I recommend you **stop reading**, go sign up for Claude Code, Gemini CLI, Aider, Codex, or some other coding agent, and try it out on a problem you know well. It'll cost about $20 / month for something useful, and maybe $100 / month for access to a state-of-the-art model. I reckon you'll be able to get surprising successes (and interesting failures) with about two hours of work.

Look Just Try Claude Code

As long as I've been in the field, automated reasoning has evolved slowly. We are used to small-% improvements achieved through clever, careful work. Claude Code breaks that pattern. Even with the limitations it has today, it can do things that seemed utterly out of reach even a year ago. Even more surprising, this capability doesn't come from some fancy solver or novel algorithm; Claude Code wasn't designed for theorem proving at all!

I think what Claude Code really points to is the bitter lesson coming for formal methods, just as it did for image recognition, language translation, and program synthesis. Just as in those fields, in the long run I think formal methods *ideas* will be essential in making AI-driven proof successful. That's because AIs will need many of the same tools as humans do to decompose, analyze, and debug proofs. But before that happens, I think we will see much of the clever, careful work we esteem rendered obsolete by AI-driven tools that have no particular domain expertise. The lesson will be bitter indeed, and many people are resisting it.

I think the result will be worth the pain, however. The reason ITP has never been widely adopted is that it is simply too cognitively demanding for most humans. Claude Code points to a future where theorem proving is *solved* - cheap, abundant, and automatic. I think that would be a good future, and if it happens, we should be ready and know what problem we want to solve next.

Look Just Try Claude Code

As long as I've been in the field, automated reasoning has evolved slowly. We are used to small-% improvements achieved through clever, careful work. Claude Code breaks that pattern. Even with the limitations it has today, it can do things that seemed utterly out of reach even a year ago. Even more surprising, this capability doesn't come from some fancy solver or novel algorithm; Claude Code wasn't designed for theorem proving at all!

I think what Claude Code really points to is the bitter lesson coming for formal methods, just as it did for image recognition, language translation, and program synthesis. Just as in those fields, in the long run I think formal methods *ideas* will be essential in making AI-driven proof successful. That's because AIs will need many of the same tools as humans do to decompose, analyze, and debug proofs. But before that happens, I think we will see much of the clever, careful work we esteem rendered obsolete by AI-driven tools that have no particular domain expertise. The lesson will be bitter indeed, and many people are resisting it.

I think the result will be worth the pain, however. The reason ITP has never been widely adopted is that it is simply too cognitively demanding for most humans. Claude Code points to a future where theorem proving is *solved* - cheap, abundant, and automatic. I think that would be a good future, and if it happens, we should be ready and know what problem we want to solve next.

Summary: Obstacles to Universality in ITP

1. Too many languages
2. Too few interaction modes
3. Too many black boxes

Summary: Obstacles to Universality in ITP

1. Too many languages
2. Too few interaction modes
3. Too many black boxes

Profound

https://dub.sh/profound

Outline

1. Level 1: contextual reasoning
2. Level 2: proof by linking
3. Level 3: quantification
4. Level 4: hierarchical detail
5. Future work
   - Dealing with dependent types
   - Bidirectional communication with existing systems

Level 1: Contextual Reasoning

Level 1: The Calculus of Structures (CoS)

- Observation: connectives are functors
  - If $A \vdash B$ then $(A \wedge F) \vdash (B \wedge F)$
  - If $A \vdash B$ then $(B \Rightarrow F) \vdash (A \Rightarrow F)$

Level 1: The Calculus of Structures (CoS)

- Observation: connectives are functors
  - If $A \vdash B$ then $(A \wedge F) \vdash (B \wedge F)$
  - If $A \vdash B$ then $(B \Rightarrow F) \vdash (A \Rightarrow F)$
  - If $A \vdash B$ then $\exists x.A \vdash \exists x.B$
  - If $A \vdash B$ then $\Box A \vdash \Box B$

Level 1: The Calculus of Structures (CoS)

- Observation: connectives are functors
  - If $A \vdash B$ then $(A \wedge F) \vdash (B \wedge F)$
  - If $A \vdash B$ then $(B \Rightarrow F) \vdash (A \Rightarrow F)$

- CoS makes use of this functoriality
  - Inference rules can operate in formula contexts:

$$\frac{A}{B} \qquad \longrightarrow \qquad \frac{\mathscr{C}\{A\}}{\mathscr{C}\{B\}}$$

  - Contexts: $(* \in \{\wedge, \vee\})$

$$\mathscr{C}\{\,\} ::= \{\,\} \mid A * \mathscr{C}\{\,\} \mid \mathscr{C}\{\,\} * B \mid A \Rightarrow \mathscr{C}\{\,\} \mid \mathscr{A}\{\,\} \Rightarrow B$$
$$\mathscr{A}\{\,\} ::= A * \mathscr{A}\{\,\} \mid \mathscr{A}\{\,\} * B \mid A \Rightarrow \mathscr{A}\{\,\} \mid \mathscr{C}\{\,\} \Rightarrow B$$

  - $\mathscr{C}\{A\}$ and $\mathscr{A}\{A\}$ replace the unique $\{\,\}$ in $\mathscr{C}\{\,\}$ or $\mathscr{A}\{\,\}$ with A.

Calculus of Structures: Summary

- Linear sequence of formulas in a derivation — no branching!
- A formula is derivable if there is a CoS derivation from a trivial theorem such as $\top$.
- Completeness: every true formula is derivable
- "Subformula" property: in $\dfrac{\mathscr{C}\{A\}}{\mathscr{C}\{B\}}$, A built from immediate subformulas of B

$$\dfrac{\top \wedge \top \wedge \top \wedge \top}{\top \wedge a \Rightarrow a \wedge \top \wedge \top}}{a \Rightarrow (\top \wedge a \wedge \top \wedge \top)}}{a \Rightarrow (\top \wedge a \wedge \top \wedge \top)}}{a \Rightarrow (a \Rightarrow a \wedge a \wedge \top \wedge \top)}}{a \Rightarrow a \Rightarrow (a \wedge a \wedge \top \wedge \top)}}{a \Rightarrow a \Rightarrow (a \wedge a \wedge \top \wedge \top)}}{a \Rightarrow (a \wedge a \wedge \top \wedge \top)}}{a \Rightarrow (a \wedge a \wedge \top \wedge c \Rightarrow c)}}{a \Rightarrow (a \Rightarrow a \Rightarrow \top \Rightarrow c) \Rightarrow c}}{(a \Rightarrow a \Rightarrow \top \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow \top \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow b \Rightarrow b \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow (b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow (a \Rightarrow b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \wedge (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}$$

CoS: Logical Rules in a Positive Context

Analogous to sequent calculus

- Initial: $$\frac{\mathscr{C}\{\mathsf{T}\}}{\mathscr{C}\{a \Rightarrow a\}}$$

- Conjunctions: $$\frac{\mathscr{C}\{(A \Rightarrow B) \wedge C\}}{\mathscr{C}\{A \Rightarrow (B \wedge C)\}} \qquad \frac{\mathscr{C}\{B \wedge (A \Rightarrow C)\}}{\mathscr{C}\{A \Rightarrow (B \wedge C)\}}$$
$$\frac{\mathscr{C}\{A \Rightarrow (B \Rightarrow C)\}}{\mathscr{C}\{(A \wedge B) \Rightarrow C\}} \qquad \frac{\mathscr{C}\{B \Rightarrow (A \Rightarrow C)\}}{\mathscr{C}\{(A \wedge B) \Rightarrow C\}}$$

- Implications: $$\frac{\mathscr{C}\{A \wedge B \Rightarrow C\}}{\mathscr{C}\{A \Rightarrow (B \Rightarrow C)\}} \qquad \frac{\mathscr{C}\{B \Rightarrow (A \Rightarrow C)\}}{\mathscr{C}\{A \Rightarrow (B \Rightarrow C)\}}$$
$$\frac{\mathscr{C}\{A \wedge (B \Rightarrow C)\}}{\mathscr{C}\{(A \Rightarrow B) \Rightarrow C\}}$$

CoS: Logical Rules in a Positive Context (contd.)

Analogous to sequent calculus

- Disjunctions:

$$\frac{\mathscr{C}\{A \Rightarrow B\}}{\mathscr{C}\{A \Rightarrow (B \vee C)\}} \qquad \frac{\mathscr{C}\{A \Rightarrow C\}}{\mathscr{C}\{A \Rightarrow (B \vee C)\}}$$

$$\frac{\mathscr{C}\{(A \Rightarrow C) \wedge (B \Rightarrow C)\}}{\mathscr{C}\{(A \vee B) \Rightarrow C\}}$$

CoS: Composition Rules in a Negative Context

No shallow analogue

- Conjunctions:

$$\frac{\mathscr{A}\{A \wedge B\}}{\mathscr{A}\{A \wedge (B \wedge C)\}} \qquad \frac{\mathscr{A}\{A \wedge C\}}{\mathscr{A}\{A \wedge (B \wedge C)\}}$$

$$\frac{\mathscr{A}\{A \wedge C\}}{\mathscr{A}\{(A \wedge B) \wedge C\}} \qquad \frac{\mathscr{A}\{B \wedge C\}}{\mathscr{A}\{(A \wedge B) \wedge C\}}$$

- Disjunctions:

$$\frac{\mathscr{A}\{(A \wedge B) \vee (A \wedge C)\}}{\mathscr{A}\{A \wedge (B \vee C)\}} \qquad \frac{\mathscr{A}\{(A \wedge C) \vee (B \wedge C)\}}{\mathscr{A}\{(A \vee B) \wedge C\}}$$

- Implications:

$$\frac{\mathscr{A}\{(A \Rightarrow B) \Rightarrow C\}}{\mathscr{A}\{A \wedge (B \Rightarrow C)\}} \qquad \frac{\mathscr{A}\{B \Rightarrow (A \wedge C)\}}{\mathscr{A}\{A \wedge (B \Rightarrow C)\}}$$

Example: S-Combinator

https://dub.sh/profound

CoS: Miscellaneous

- Units and Simplification:

$$\frac{\mathscr{C}\{\top\}}{\mathscr{C}\{A \Rightarrow \top\}} \qquad \frac{\mathscr{C}\{B\}}{\mathscr{C}\{\top \Rightarrow B\}} \qquad \frac{\mathscr{C}\{\top\}}{\mathscr{C}\{\bot \Rightarrow B\}}$$

$$\frac{\mathscr{C}\{A\}}{\mathscr{C}\{\top \wedge A\}} \qquad \frac{\mathscr{C}\{A\}}{\mathscr{C}\{A \wedge \top\}} \qquad \frac{\mathscr{A}\{A\}}{\mathscr{A}\{\bot \vee A\}} \qquad \frac{\mathscr{A}\{A\}}{\mathscr{A}\{A \vee \bot\}}$$

$$\frac{\mathscr{A}\{\bot\}}{\mathscr{A}\{\bot \wedge A\}} \qquad \frac{\mathscr{A}\{\bot\}}{\mathscr{A}\{A \wedge \bot\}} \qquad \frac{\mathscr{C}\{\top\}}{\mathscr{C}\{\top \vee A\}} \qquad \frac{\mathscr{C}\{\top\}}{\mathscr{C}\{A \vee \top\}}$$

- Contraction, Cut:

$$\frac{\mathscr{C}\{A \Rightarrow A \Rightarrow C\}}{\mathscr{C}\{A \Rightarrow C\}} \qquad \frac{\mathscr{C}\{A \wedge (A \Rightarrow C)\}}{\mathscr{C}\{C\}}$$

CoS: Meta-Theorems

Soundness

If $\begin{array}{c} A \\ \vdots \\ C \end{array}$ then in $A \vdash C$ is provable.

Completeness

If $A_1, \ldots, A_n \vdash C$ is provable, then: $A_1 \Rightarrow \cdots \Rightarrow A_n \Rightarrow C$ $\begin{array}{c} \top \\ \vdots \\ \cdot \end{array}$

Interlude: Exporting CoS to Other Systems

Case of Rocq

- Challenge: Given a CoS derivation $\begin{array}{c} A \\ \vdots \\ C \end{array}$, fill in the details of:

```
Goal A -> C.                              (* A, C : Prop. *)
Proof. (* insert proof here *) Qed.
```

Interlude: Exporting CoS to Other Systems

Case of Rocq

- Challenge: Given a CoS derivation $\begin{array}{c} A \\ \vdots \\ C \end{array}$, fill in the details of:

```
Goal A -> C.                                (* A, C : Prop. *)
Proof. (* insert proof here *) Qed.
```

- Two main styles:
  - Shallow embedding: translate CoS proof rules to Rocq tacticals.
  - Deep embedding: represent the CoS proof as a Rocq data structure, and prove (as a meta-theorem in Rocq) it has a sound reflection function.

Interlude: Shallow Embedding of CoS Rules

- Translating the rule instance $\dfrac{\mathscr{C}\{A\}}{\mathscr{C}\{C\}}$ requires:
  - Showing the entailment $A \vdash C$, and
  - Transporting the entailment to $\mathscr{C}\{\}$.

Interlude: Shallow Embedding of CoS Rules

- Translating the rule instance $\frac{\mathscr{C}\{A\}}{\mathscr{C}\{C\}}$ requires:
  - Showing the entailment $A \vdash C$, and
  - Transporting the entailment to $\mathscr{C}\{\}$.
- Transport combinators:

```
Theorem and_l {A B C : Prop} : (A -> B) -> ((A /\ C) -> (B /\ C)).
Theorem imp_r {A B C : Prop} : (A -> B) -> ((C -> A) -> (C -> B)).

Check (imp_r (and_l _)).
  (* imp_r (and_l ?y) : (?C1 -> ?A /\ ?C2) -> (?C1 -> ?B /\ ?C2)
     where ?y : [ |- ?A -> ?B] *)
```

Interlude: Shallow Embedding of CoS Rules

- Translating the rule instance $\frac{\mathscr{C}\{A\}}{\mathscr{C}\{C\}}$ requires:
  - Showing the entailment $A \vdash C$, and
  - Transporting the entailment to $\mathscr{C}\{\}$.
- Transport combinators:

```
Theorem and_l {A B C : Prop} : (A -> B) -> ((A /\ C) -> (B /\ C)).
Theorem imp_r {A B C : Prop} : (A -> B) -> ((C -> A) -> (C -> B)).

Check (imp_r (and_l _)).
  (* imp_r (and_l ?y) : (?C1 -> ?A /\ ?C2) -> (?C1 -> ?B /\ ?C2)
     where ?y : [ |- ?A -> ?B] *)
```

- Rules:

```
Theorem g_imp_and_l {A B C : Prop} : (A -> B) /\ C -> (A -> B /\ C).

Check (imp_r (and_l g_imp_and_l)).
 (* imp_r (and_l g_imp_and_l)
    : (?C -> ((?A -> ?B) /\ ?D) /\ ?E) -> ?C -> (?A -> ?B /\ ?D) /\ ?E *)
```

Interlude: Shallow Embedding of CoS Rules

Final assembly

```
Theorem imp_r {A B C : Prop}       : (A -> B) -> (C -> A) -> (C -> B).
Theorem g_imp_and_l {A B C : Prop} : (A -> B) /\ C -> (A -> B /\ C).
Theorem g_imp_imp_r {A B C : Prop} : (B -> A -> C) -> (A -> B -> C).
Theorem g_init {A : Prop}          : True -> (A -> A).
Theorem s_imp_true {A : Prop}      : True -> (A -> True).

Goal forall (A B : Prop), A -> B -> A.
Proof.
  intros A B.
  refine (g_imp_imp_r _).  (* B -> A -> A *)
  refine (imp_r g_init _). (* B -> True *)
  refine (s_imp_true _).   (* True *)
  constructor.
Qed.
```

Level 2: Linking

Level 2: Linking

- CoS rules are more verbose than even sequent rules

Level 2: Linking

- CoS rules are more verbose than even sequent rules
- Goal: use the freedom of CoS and avoid the tedium

Level 2: Linking

- CoS rules are more verbose than even sequent rules
- Goal: use the freedom of CoS and avoid the tedium
- Linking:
  - Each link joins two unrelated subformulas
  - The user indicates the ends of the link
  - The system figures out how to resolve the link

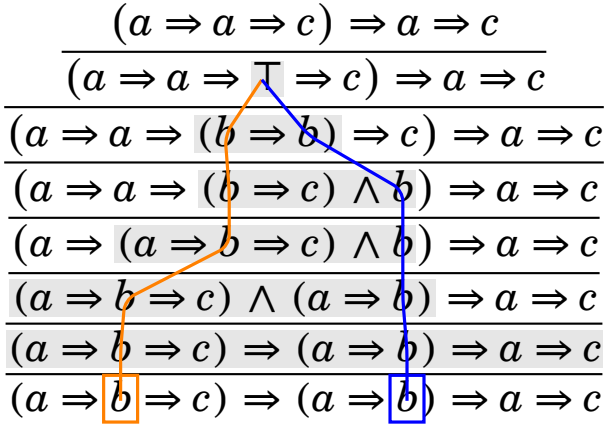
Linking: Indicating and Resolving

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(a \Rightarrow a \Rightarrow c) \Rightarrow a \Rightarrow c}{(a \Rightarrow a \Rightarrow \top \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow (b \Rightarrow b) \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow (b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow (a \Rightarrow b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \wedge (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}$$

Linking: Indicating and Resolving

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(a \Rightarrow a \Rightarrow c) \Rightarrow a \Rightarrow c}{(a \Rightarrow a \Rightarrow \top \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow (b \Rightarrow b) \Rightarrow c) \Rightarrow a \Rightarrow c}}{(a \Rightarrow a \Rightarrow (b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow (a \Rightarrow b \Rightarrow c) \wedge b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \wedge (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow b \Rightarrow c) \Rightarrow (a \Rightarrow b) \Rightarrow a \Rightarrow c}}{(a \Rightarrow \boxed{b} \Rightarrow c) \Rightarrow (a \Rightarrow \boxed{b}) \Rightarrow a \Rightarrow c}$$

Linking: Indicating and Resolving



Linking: Indicating

- A *linked* formula has one of the following two shapes:

$$\mathscr{C}_0\{\mathscr{C}_1\{\underbrace{A\}\Rightarrow\mathscr{C}_2\{B}\}\}\qquad\text{or}\qquad\mathscr{A}_0\{\mathscr{C}_1\{\underbrace{A\}\wedge\mathscr{C}_2\{B}\}\}$$

Linking: Indicating

- A *linked* formula has one of the following two shapes:

$$\mathscr{C}_0\{\mathscr{C}_1\{\underbrace{A\}\Rightarrow\mathscr{C}_2\{B}\}\}\qquad\text{or}\qquad\mathscr{A}_0\{\mathscr{C}_1\{\underbrace{A\}\wedge\mathscr{C}_2\{B}\}\}$$

- Link inititation can be written as inference rules:

$$\frac{\mathscr{C}_0\{\mathscr{C}_1\{\underbrace{A\}\Rightarrow\mathscr{C}_2\{B}\}\}}{\mathscr{C}_0\{\mathscr{C}_1\{A\}\Rightarrow\mathscr{C}_2\{B\}\}}\qquad\frac{\mathscr{A}_0\{\mathscr{C}_1\{\underbrace{A\}\wedge\mathscr{C}_2\{B}\}\}}{\mathscr{A}_0\{\mathscr{C}_1\{A\}\wedge\mathscr{C}_2\{B\}\}}$$

Linking: Link Resolution

- Create variants of the CoS rules that operate on linked formulas
- In each case the link is shorter in the premise
- Some cases:

$$\frac{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\}) \wedge C\}}{\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\} \wedge C\}} \qquad \frac{\mathscr{C}_0\{A \wedge (\mathscr{C}_1\{B\} \Rightarrow \mathscr{C}_2\{C\})\}}{\mathscr{C}_0\{(A \Rightarrow \mathscr{C}_1\{B\}) \Rightarrow \mathscr{C}_2\{C\}\}}$$

$$\frac{\mathscr{A}_0\{(\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\}) \Rightarrow C\}}{\mathscr{A}_0\{\mathscr{C}_1\{A\} \wedge (\mathscr{C}_2\{B\} \Rightarrow C)\}}$$

Link Resolution: Choices

$$\frac{\mathscr{C}_0\{B \Rightarrow \mathscr{C}_1\{A\} \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad \text{or} \quad \frac{\mathscr{C}_0\{((\mathscr{C}_1\{A\} \wedge B) \Rightarrow \mathscr{C}_2\{C\}) \wedge D\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad ?$$

Link Resolution: Choices

$$\frac{\mathscr{C}_0\{B \Rightarrow \mathscr{C}_1\{A\} \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad \text{or} \quad \frac{\mathscr{C}_0\{((\mathscr{C}_1\{A\} \wedge B) \Rightarrow \mathscr{C}_2\{C\}) \wedge D\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad ?$$

- Most of the time there is a reasonable choice
  - One of the rules is invertible, in which case do it first
  - If both rules are invertible, the choice does not matter
  - If both rules are non-invertible and in positively signed contexts, the choice turns out not to matter

Link Resolution: Choices

$$\frac{\mathscr{C}_0\{B \Rightarrow \mathscr{C}_1\{A\} \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad \text{or} \quad \frac{\mathscr{C}_0\{((\mathscr{C}_1\{A\} \wedge B) \Rightarrow \mathscr{C}_2\{C\}) \wedge D\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad ?$$

- Most of the time there is a reasonable choice
  - One of the rules is invertible, in which case do it first
  - If both rules are invertible, the choice does not matter
  - If both rules are non-invertible and in positively signed contexts, the choice turns out not to matter
- There are critical pairs for negatively signed contexts, i.e., compositions

Link Resolution: Choices

$$\frac{\mathscr{C}_0\{B \Rightarrow \mathscr{C}_1\{A\} \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad \text{or} \quad \frac{\mathscr{C}_0\{((\mathscr{C}_1\{A\} \wedge B) \Rightarrow \mathscr{C}_2\{C\}) \wedge D\}}{\mathscr{C}_0\{(\mathscr{C}_1\{A\} \wedge B) \Rightarrow (\mathscr{C}_2\{C\} \wedge D)\}} \quad ?$$

- Most of the time there is a reasonable choice
  - One of the rules is invertible, in which case do it first
  - If both rules are invertible, the choice does not matter
  - If both rules are non-invertible and in positively signed contexts, the choice turns out not to matter
- There are critical pairs for negatively signed contexts, i.e., compositions

$$\frac{\mathscr{A}\{(A \Rightarrow (B \wedge C)) \vee D\}}{\mathscr{A}\{((A \Rightarrow B) \wedge C) \vee D\}}{\mathscr{A}\{(A \Rightarrow B) \wedge (C \vee D)\}} \quad \text{vs} \quad \frac{\mathscr{A}\{A \Rightarrow ((B \wedge C) \vee D)\}}{\mathscr{A}\{A \Rightarrow (B \wedge (C \vee D))\}}{\mathscr{A}\{(A \Rightarrow B) \wedge (C \vee D)\}}$$

Link Resolution: Directional Links

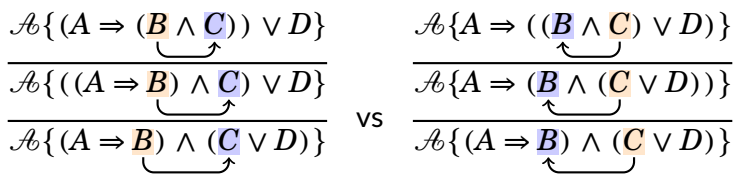
- The situation is even more complicated with quantifiers
- Use the order of links to determine the nesting order

$$\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\}\} \qquad \text{or} \qquad \mathscr{A}_0\{\mathscr{C}_1\{A\} \wedge \mathscr{C}_2\{B\}\}$$

- Intuition is to insert the source into the destination

$$\mathscr{C}_0\{\mathscr{C}_2\{\mathscr{C}_1\{A \Rightarrow B\}\}\}$$
$$\vdots$$
$$\mathscr{C}_0\{\mathscr{C}_2\{\mathscr{C}_1\{A\} \Rightarrow B\}\}$$
$$\vdots$$
$$\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\}\}$$

Link Resolution: Directional Links



Link Resolution: Finishing

- When a link has length 0, it can be removed

$$\frac{\mathscr{C}\{\top\}}{\mathscr{C}\{A \Rightarrow A\}} \qquad \frac{\mathscr{C}\{A \Rightarrow B\}}{\mathscr{C}\{A \Rightarrow B\}} \qquad \frac{\mathscr{A}\{A \wedge B\}}{\mathscr{A}\{A \wedge B\}} \qquad (A \neq B)$$

- The formula can be simplified with respect to $\top$.

$$\frac{\mathscr{C}\{A\}}{\mathscr{C}\{A \wedge \top\}} \qquad \frac{\mathscr{C}\{\top\}}{\mathscr{C}\{A \Rightarrow \top\}} \qquad \text{etc.}$$

Profound

https://dub.sh/profound

Level 3: First-Order

Level 3: First-Order Quantification

- It is common to see quantification done carelessly:

$$
\begin{aligned}
A, B, \ldots &::= \cdots \mid \forall x. A \mid \exists x. A \\
\mathscr{C}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{C}\{\,\} \mid \exists x.\, \mathscr{C}\{\,\} \\
\mathscr{A}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{A}\{\,\} \mid \exists x.\, \mathscr{A}\{\,\}
\end{aligned}
$$

Level 3: First-Order Quantification

- It is common to see quantification done carelessly:

$$
\begin{aligned}
A, B, \ldots &::= \cdots \mid \forall x.\, A \mid \exists x.\, A \\
\mathscr{C}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{C}\{\,\} \mid \exists x.\, \mathscr{C}\{\,\} \\
\mathscr{A}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{A}\{\,\} \mid \exists x.\, \mathscr{A}\{\,\}
\end{aligned}
$$

- Problem: $\mathscr{C}\{A\}$ interpreted as capturing the free variables of A

Level 3: First-Order Quantification

- It is common to see quantification done carelessly:

$$
\begin{aligned}
A, B, \ldots &::= \cdots \mid \forall x. A \mid \exists x. A \\
\mathscr{C}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{C}\{\,\} \mid \exists x.\, \mathscr{C}\{\,\} \\
\mathscr{A}\{\,\} &::= \cdots \mid \forall x.\, \mathscr{A}\{\,\} \mid \exists x.\, \mathscr{A}\{\,\}
\end{aligned}
$$

- Problem: $\mathscr{C}\{A\}$ interpreted as capturing the free variables of A
- Problematic to implement, formalize the meta-theory, export

First-Order Quantification

- Alternative: raising

$$\mathscr{C}_{\Gamma,x}\{\,\} ::= \cdots \mid \forall x.\, \mathscr{C}_{\Gamma}\{\,\} \mid \exists x.\, \mathscr{C}_{\Gamma}\{\,\}$$
$$\mathscr{A}_{\Gamma,x}\{\,\} ::= \cdots \mid \forall x.\, \mathscr{A}_{\Gamma}\{\,\} \mid \exists x.\, \mathscr{A}_{\Gamma}\{\,\}$$

- Intuition: if $\Gamma \vdash A : \mathsf{prop}$ then $\mathscr{C}_{\Gamma}\{A\}$ is well-formed:

$$\forall u.\, \mathscr{C}_{\Gamma,x}\{A\} \;=\; \forall u.\, \mathscr{C}_{\Gamma}\{[u/x]A\}$$

Interlude: Representing Contexts

In Rocq

```
Inductive cx : list Type -> Type :=
  | Hole        : cx nil
  | C_AndL Ts   : cx Ts -> Prop -> cx Ts
  | C_AndR Ts   : Prop -> cx Ts -> cx Ts
  | C_OrL Ts    : cx Ts -> Prop -> cx Ts
  | C_OrR Ts    : Prop -> cx Ts -> cx Ts
  | C_ImpL Ts   : ax Ts -> Prop -> cx Ts
  | C_ImpR Ts   : Prop -> cx Ts -> cx Ts
  | C_AllD A Ts : (A -> cx Ts) -> cx (A :: Ts)
  | C_ExD A Ts  : (A -> cx Ts) -> cx (A :: Ts).
and ax : list Type -> Type := ...
```

Interlude: Representing Raised Formulas

In Rocq

```
Fixpoint raise (Ts : list Type) (U : Type) : Type :=
  match Ts with
  | nil => U
  | A :: Ts => A -> raise Ts U
  end.
Notation "Ts ▷ U" := (raise Ts U).
```

Interlude: Representing Raised Formulas

In Rocq

```
Fixpoint raise (Ts : list Type) (U : Type) : Type :=
  match Ts with
  | nil => U
  | A :: Ts => A -> raise Ts U
  end.
Notation "Ts ▷ U" := (raise Ts U).
```

```
Fixpoint cx_place Ts (cx : cx Ts) : (Ts ▷ Prop) -> Prop :=
  match cx with
  | Hole        => fun p => p
  | AndL _ cx q => fun p => cx{{ p }} /\ q
  | ...
  | AllD A _ cx => fun p => forall (x : A), (cx x){{ p x }}
  | ExD A _ cx  => fun p => exists (x : A), (cx x){{ p x }}
  end
where "Cx {{ P }}" := (@cx_place _ Cx P).
```

Interlude: Building Raised Formulas

In Rocq

```
Fixpoint raised_and (Ts : list Type)
    : (Ts ▷ Prop) -> (Ts ▷ Prop) -> (Ts ▷ Prop) :=
  match Ts with
  | nil     => (fun p q => p /\ q)
  | A :: Ts => (fun (p q : A -> (Ts ▷ Prop)) (x : A) =>
                 raised_and Ts (p x) (q x))
  end.
Notation "f ∧ g" := (raised_and _ f g).
```

Interlude: Building Raised Formulas

In Rocq

```
Fixpoint raised_and (Ts : list Type)
    : (Ts ▷ Prop) -> (Ts ▷ Prop) -> (Ts ▷ Prop) :=
  match Ts with
  | nil     => (fun p q => p /\ q)
  | A :: Ts => (fun (p q : A -> (Ts ▷ Prop)) (x : A) =>
                 raised_and Ts (p x) (q x))
  end.
Notation "f ∧ g" := (raised_and _ f g).
```

```
Fixpoint raised_forall (A : Type) (Ts : list Type)
    : (A -> (Ts ▷ Prop)) -> (Ts ▷ Prop) :=
  match Ts with
  | nil     => (fun p => forall (x : A), p x)
  | B :: Ts => (fun p (x : B) =>
                 raised_forall A Ts (fun u => p u x))
  end.
Notation "∀ x .. y , p" :=
  (raised_forall _ _ (fun x => .. (raised_forall_all _ _ (fun y => p)) ..))
  (x binder).
```

Linking: Quantifier Rules

- Extrusion:

$$\frac{\mathscr{C}_0\{\forall x.\,(\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\})\}}{\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \forall x.\,\mathscr{C}_2\{B\}\}} \qquad \frac{\mathscr{A}_0\{\forall x.\,(\mathscr{C}_1\{A\} \wedge \mathscr{C}_2\{B\})\}}{\mathscr{A}_0\{\mathscr{C}_1\{A\} \wedge \forall x.\,\mathscr{C}_2\{B\}\}}$$

  etc.

Linking: Quantifier Rules

- Extrusion:

$$\frac{\mathscr{C}_0\{\forall x.\ (\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\})\}}{\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \forall x.\ \mathscr{C}_2\{B\}\}} \qquad \frac{\mathscr{A}_0\{\forall x.\ (\mathscr{C}_1\{A\} \wedge \mathscr{C}_2\{B\})\}}{\mathscr{A}_0\{\mathscr{C}_1\{A\} \wedge \forall x.\ \mathscr{C}_2\{B\}\}}$$

  etc.

- Instantiation:

$$\frac{\mathscr{C}\{[t/x]A\}}{\mathscr{C}\{\exists x.A\}} \qquad \frac{\mathscr{A}\{[t/x]A\}}{\mathscr{A}\{\forall x.A\}}$$

Linking: Quantifier Rules

- Extrusion:

$$\frac{\mathscr{C}_0\{\forall x.\ (\mathscr{C}_1\{A\} \Rightarrow \mathscr{C}_2\{B\})\}}{\mathscr{C}_0\{\mathscr{C}_1\{A\} \Rightarrow \forall x.\ \mathscr{C}_2\{B\}\}} \qquad \frac{\mathscr{A}_0\{\forall x.\ (\mathscr{C}_1\{A\} \wedge \mathscr{C}_2\{B\})\}}{\mathscr{A}_0\{\mathscr{C}_1\{A\} \wedge \forall x.\ \mathscr{C}_2\{B\}\}}$$

  etc.

- Instantiation:

$$\frac{\mathscr{C}\{[t/x]A\}}{\mathscr{C}\{\exists x.A\}} \qquad \frac{\mathscr{A}\{[t/x]A\}}{\mathscr{A}\{\forall x.A\}}$$

- Simplification:

$$\frac{\mathscr{C}\{\top\}}{\mathscr{C}\{\forall x.\ \top\}}$$

Linking: Predicates

- Initial:

$$\frac{\mathscr{C}\{s_1 \doteq t_1 \wedge \cdots \wedge s_n \doteq t_n\}}{\mathscr{C}\{\mathsf{a}\, s_1 \cdots s_n \Rightarrow \mathsf{a}\, t_1 \cdots t_n\}}$$

Linking: Predicates

- Initial:

$$\frac{\mathscr{C}\{s_1 \doteq t_1 \wedge \cdots \wedge s_n \doteq t_n\}}{\mathscr{C}\{\mathsf{a}\, s_1 \cdots s_n \Rightarrow \mathsf{a}\, t_1 \cdots t_n\}}$$

- Equality simplification:

$$\frac{\mathscr{C}\{s_1 \doteq t_1 \wedge \cdots \wedge s_n \doteq t_n\}}{\mathscr{C}\{\mathsf{f}\, s_1 \cdots s_n \doteq \mathsf{f}\, t_1 \cdots t_n\}} \qquad \frac{\mathscr{C}\{\mathsf{T}\}}{\mathscr{C}\{s \doteq s\}}$$

Linking: Predicates

- Initial:

$$\frac{\mathscr{C}\{s_1 \doteq t_1 \wedge \cdots \wedge s_n \doteq t_n\}}{\mathscr{C}\{\mathsf{a}\, s_1 \cdots s_n \Rightarrow \mathsf{a}\, t_1 \cdots t_n\}}$$

- Equality simplification:

$$\frac{\mathscr{C}\{s_1 \doteq t_1 \wedge \cdots \wedge s_n \doteq t_n\}}{\mathscr{C}\{\mathsf{f}\, s_1 \cdots s_n \doteq \mathsf{f}\, t_1 \cdots t_n\}} \qquad \frac{\mathscr{C}\{\top\}}{\mathscr{C}\{s \doteq s\}}$$

- Rewriting:

$$\frac{\mathscr{C}\{A\{t\}\}}{\mathscr{C}\{s \doteq t \Rightarrow A\{s\}\}} \qquad \frac{\mathscr{C}\{A\{s\}\}}{\mathscr{C}\{s \doteq t \Rightarrow A\{t\}\}}$$

Linking: Instantiation Heuristics

- Guess instances:

$$\frac{\mathscr{C}\{\forall x.\, \mathscr{C}_1\{x \doteq t\} \Rightarrow [t/x]A\}}{\mathscr{C}\{\forall x.\, \mathscr{C}_1\{x \doteq t\} \Rightarrow A\}} \qquad \frac{\mathscr{C}\{\exists x.\, \mathscr{C}_1\{x \doteq t\} \wedge [t/x]A\}}{\mathscr{C}\{\exists x.\, \mathscr{C}_1\{x \doteq t\} \wedge A\}}$$

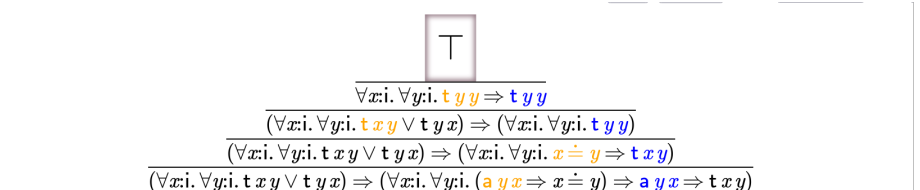
Linking: Instantiation Heuristics

- Guess instances:

$$\frac{\mathscr{C}\{\forall x.\, \mathscr{C}_1\{x \doteq t\} \Rightarrow [t/x]A\}}{\mathscr{C}\{\forall x.\, \mathscr{C}_1\{x \doteq t\} \Rightarrow A\}} \qquad \frac{\mathscr{C}\{\exists x.\, \mathscr{C}_1\{x \doteq t\} \wedge [t/x]A\}}{\mathscr{C}\{\exists x.\, \mathscr{C}_1\{x \doteq t\} \wedge A\}}$$

- Other approaches are possible
  - Unification
  - Theory reasoning
  - Issue: export to other provers

Level 4: Clutter Management


⊤
∀x:i. ∀y:i. t y y ⇒ t y y
(∀x:i. ∀y:i. t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. t y y)
(∀x:i. ∀y:i. t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. x ≐ y ⇒ t x y)
(∀x:i. ∀y:i. t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. (a y x ⇒ x ≐ y) ⇒ a y x ⇒ t x y)
(∀x:i. ∀y:i. t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. a x y ⇒ a y x ⇒ x ≐ y) ⇒ (∀x:i. ∀y:i. a x y ⇒ a y x ⇒ t x y)
t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. a x y ⇒ a y x ⇒ x ≐ y) ⇒ (∀x:i. ∀y:i. t x y ⇒ a x y) ⇒ (∀x:i. ∀y:i. a x y ⇒ t y x
7y:i. t x y ∨ t y x) ⇒ (∀x:i. ∀y:i. a x y ⇒ a y x ⇒ x ≐ y) ⇒ (∀x:i. ∀y:i. t x y ⇒ a x y) ⇒ (∀x:i. ∀y:i. a x y ⇒ t

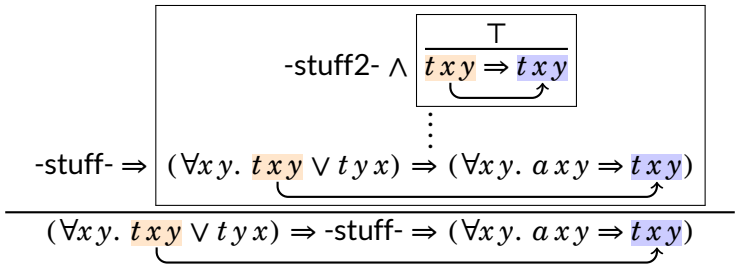
Open Deduction

- We actually use open deduction instead of CoS
- Combined syntax for formulas, sequents, and proofs

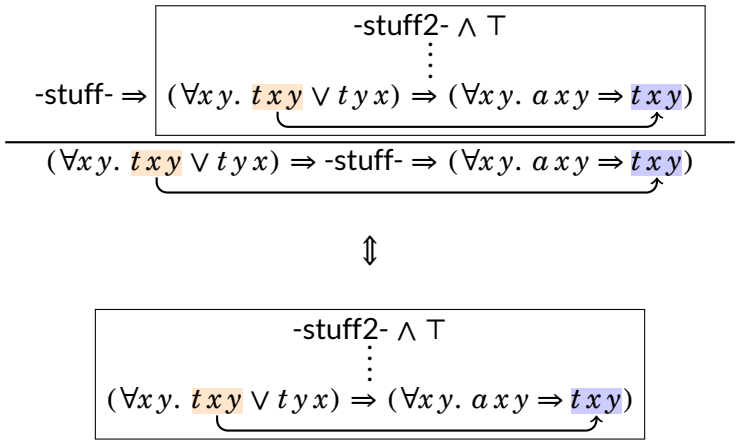
$$\mathscr{D} ::= A \mid \mathscr{D}_1 * \mathscr{D}_2 \mid \boxed{\frac{\mathscr{D}_1}{\mathscr{D}_2}} \mid \frac{\boxed{\mathscr{D}_1}}{\boxed{\mathscr{D}_2}}$$

- Contexts are suitably generalized as well

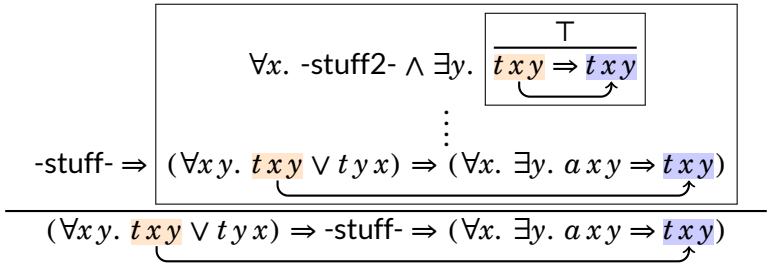
Open Deduction as a Hierarchy



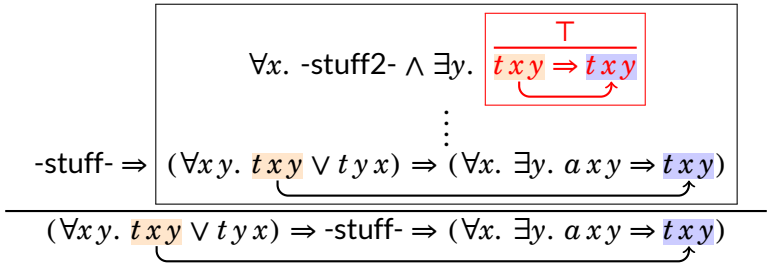
Zooming


-stuff2- ∧ ⊤
-stuff- ⇒ (∀x y. t x y ∨ t y x) ⇒ (∀x y. a x y ⇒ t x y)
(∀x y. t x y ∨ t y x) ⇒ -stuff- ⇒ (∀x y. a x y ⇒ t x y)
-stuff2- ∧ ⊤
(∀x y. t x y ∨ t y x) ⇒ (∀x y. a x y ⇒ t x y)

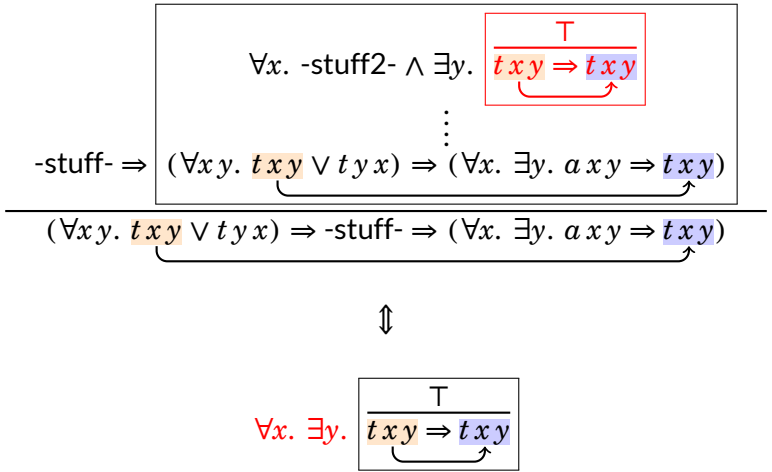
Zooming into Scopes



Zooming into Scopes



Zooming into Scopes



Launching and Linking

$$(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)$$

Launching and Linking

$(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow$ -stuff- $\Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)$

Launching and Linking

$$\frac{\text{-stuff-} \Rightarrow \boxed{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}}{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}$$

Launching and Linking

$$\frac{\text{-stuff-} \wedge F \Rightarrow \boxed{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}}{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \wedge F \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}$$

Launching and Linking

$$\frac{\text{-stuff-} \wedge F \Rightarrow \boxed{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}}{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \wedge F \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}$$

Launching and Linking

$$\frac{\text{-stuff-} \Rightarrow \boxed{F \Rightarrow (\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}}{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \wedge F \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}$$

Launching, Stopping, and Linking

$$\frac{\text{-stuff-} \Rightarrow F \Rightarrow (\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}{\text{-stuff-} \Rightarrow \boxed{F \Rightarrow (\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}}$$

$$\frac{\text{-stuff-} \Rightarrow F \Rightarrow (\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}{(\forall x\, y.\ t\, x\, y \vee t\, y\, x) \Rightarrow \text{-stuff-} \wedge F \Rightarrow (\forall x\, y.\ a\, x\, y \Rightarrow t\, x\, y)}$$

Future Work

Additional Universal Interactions

- Additional structural operations
  - Deep re-organization: splitting
  - Refactoring using substitutions

Additional Universal Interactions

- Additional structural operations
  - Deep re-organization: splitting
  - Refactoring using substitutions
- Dealing with lemmas
  - Explicit cuts
  - Appealing to previously proved theorems
  - Searching for lemmas

Additional Universal Interactions

- Additional structural operations
  - Deep re-organization: splitting
  - Refactoring using substitutions
- Dealing with lemmas
  - Explicit cuts
  - Appealing to previously proved theorems
  - Searching for lemmas
- History and persistence
  - Forking histories
  - Linking to the past

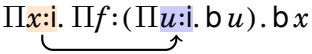
Linking for Type Theory

Highly speculative

$$\Pi x{:}\mathsf{i}.\ \Pi f : (\Pi u{:}\mathsf{i}.\ \mathsf{b}\,u).\ \mathsf{b}\,x$$

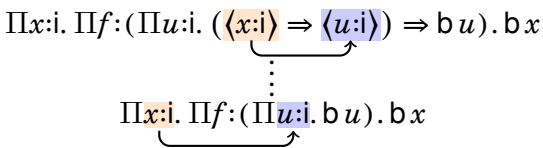
Linking for Type Theory

Highly speculative



Linking for Type Theory

Highly speculative



Linking for Type Theory: via Realizability

$$[\![\alpha]\!]\, M \triangleq \langle M{:}\alpha \rangle$$
$$[\![\Pi x{:}A.\, B]\!]\, M \triangleq \forall x.\, [\![A]\!]\, x \Rightarrow [\![B]\!]\, (M\, x)$$

Linking for Type Theory: via Realizability

$$[\![\alpha]\!]\, M \triangleq \langle M{:}\alpha \rangle$$
$$[\![\Pi x{:}A.\, B]\!]\, M \triangleq \forall x.\, [\![A]\!]\, x \Rightarrow [\![B]\!]\, (M\, x)$$

$$\forall x.\, \langle x{:}\mathsf{i} \rangle \Rightarrow \forall f.\, (\forall u.\, \langle u{:}\mathsf{i} \rangle \Rightarrow \langle (f\, u){:}\mathsf{b}\, u \rangle) \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle$$

Linking for Type Theory: via Realizability

$$\llbracket \alpha \rrbracket\, M \triangleq \langle M{:}\alpha \rangle$$
$$\llbracket \Pi x{:}A.\, B \rrbracket\, M \triangleq \forall x.\, \llbracket A \rrbracket\, x \Rightarrow \llbracket B \rrbracket\, (M\, x)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\forall x.\, \forall f.\, \exists z.\, (f\, x) \doteq z \wedge (\mathsf{b}\, x) \doteq (\mathsf{b}\, x)}{\forall x.\, \forall f.\, \langle (f\, x){:}(\mathsf{b}\, x) \rangle \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle}}{\forall x.\, \forall f.\, \langle (f\, x){:}(\mathsf{b}\, x) \rangle \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle}}{\forall x.\, \forall f.\, (\forall u.\, x \doteq u \wedge \mathsf{i} \doteq \mathsf{i} \Rightarrow \langle (f\, u){:}\mathsf{b}\, u \rangle) \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle}}{\forall x.\, \forall f.\, (\forall u.\, (\langle x{:}\mathsf{i} \rangle \Rightarrow \langle u{:}\mathsf{i} \rangle) \Rightarrow \langle (f\, u){:}\mathsf{b}\, u \rangle) \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle}$$

$$\vdots$$

$$\forall x.\, \langle x{:}\mathsf{i} \rangle \Rightarrow \forall f.\, (\forall u.\, \langle u{:}\mathsf{i} \rangle \Rightarrow \langle (f\, u){:}\mathsf{b}\, u \rangle) \Rightarrow \exists z.\, \langle z{:}\mathsf{b}\, x \rangle$$

Linking for Type Theory: Future

- Unfortunately, the “realizability interpretation” is unsound

Linking for Type Theory: Future

- Unfortunately, the “realizability interpretation” is unsound except for canonical terms

Linking for Type Theory: Future

- Unfortunately, the "realizability interpretation" is unsound except for canonical terms
- Better alternative: try to do it within type theory itself
  - Might need to generalize the syntax of "types"
  - Would need to split Π into a binder and a type assumption
  - The binder would stay put
  - The type assumption would move via linking

Linking for Type Theory: Future

- Unfortunately, the "realizability interpretation" is unsound except for canonical terms
- Better alternative: try to do it within type theory itself
  - Might need to generalize the syntax of "types"
  - Would need to split Π into a binder and a type assumption
  - The binder would stay put
  - The type assumption would move via linking
- Proposed system
  - Easy to show completeness
  - Hard to prove sound – generalized "type" derivations very hard to map to ordinary typing derivations via natural deduction, e.g.

Definitions and Notations

- Definition unfolding: double click

Definitions and Notations

- Definition unfolding: double click
- Notations: if they are functorial, can be incorporated automatically

Definitions and Notations

- Definition unfolding: double click
- Notations: if they are functorial, can be incorporated automatically
  - E.g., $\neg A \triangleq A \Rightarrow \bot$
    - If $A \vdash B$ then $\neg B \vdash \neg A$
    - This means $\neg$ has an $\mathscr{A}\{\}$-style CoS/linking rule
    - Also need to specify what $\neg\top$ and $\neg\bot$ simplify as

Definitions and Notations

- Definition unfolding: double click
- Notations: if they are functorial, can be incorporated automatically
  - E.g., $\neg A \triangleq A \Rightarrow \bot$
    - If $A \vdash B$ then $\neg B \vdash \neg A$
    - This means $\neg$ has an $\mathscr{A}\{\}$-style CoS/linking rule
    - Also need to specify what $\neg\top$ and $\neg\bot$ simplify as
  - Sometimes notations are not functorial
    - Simple example: $A \equiv B \triangleq (A \Rightarrow B) \wedge (B \Rightarrow A)$
    - $A \vdash B$ does not mean $A \equiv C \vdash B \equiv C$
    - Probably beter here to consider rewrite-stye linking, as with $\doteq$

Induction and Circular Reasoning

- Induction:

Induction and Circular Reasoning

- Induction:
  - Abella-style sized relations reasonably simple

Induction and Circular Reasoning

- Induction:
  - Abella-style sized relations reasonably simple
  - Induction invariants in CoS: open problem?

Induction and Circular Reasoning

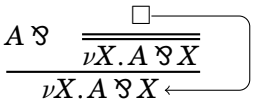
- Induction:
  - Abella-style sized relations reasonably simple
  - Induction invariants in CoS: open problem?
- Circular reasoning:

Induction and Circular Reasoning

- Induction:
  - Abella-style sized relations reasonably simple
  - Induction invariants in CoS: open problem?
- Circular reasoning:
  - Easy: indicating cycles with pointing devices

Induction and Circular Reasoning

- Induction:
  - Abella-style sized relations reasonably simple
  - Induction invariants in CoS: open problem?
- Circular reasoning:
  - Easy: indicating cycles with pointing devices
  - Hard: making sense of back edges crossing context boundaries



Bidirectional Communication with Verifiers

- Exporting proofs: already covered

Bidirectional Communication with Verifiers

- Exporting proofs: already covered
- Importing proofs?

Bidirectional Communication with Verifiers

- Exporting proofs: already covered
- Importing proofs?
  - Every shallow proof is (easily converted to) a deep proof

Bidirectional Communication with Verifiers

- Exporting proofs: already covered
- Importing proofs?
  - Every shallow proof is (easily converted to) a deep proof
  - But: too many proof languages!

Bidirectional Communication with Verifiers

- Exporting proofs: already covered
- Importing proofs?
  - Every shallow proof is (easily converted to) a deep proof
  - But: too many proof languages!
- Related problem: adding a new verifier?

Bidirectional Communication with Verifiers

- Exporting proofs: already covered
- Importing proofs?
  - Every shallow proof is (easily converted to) a deep proof
  - But: too many proof languages!
- Related problem: adding a new verifier?
- Dedukti?

Conclusion

The Future of Interactive Theorem Proving Interfaces?

https://dub.sh/profound/slides