Foundational proof certificates in first-order logic

Zakaria Chihani, Dale Miller, and Fabien Renaud

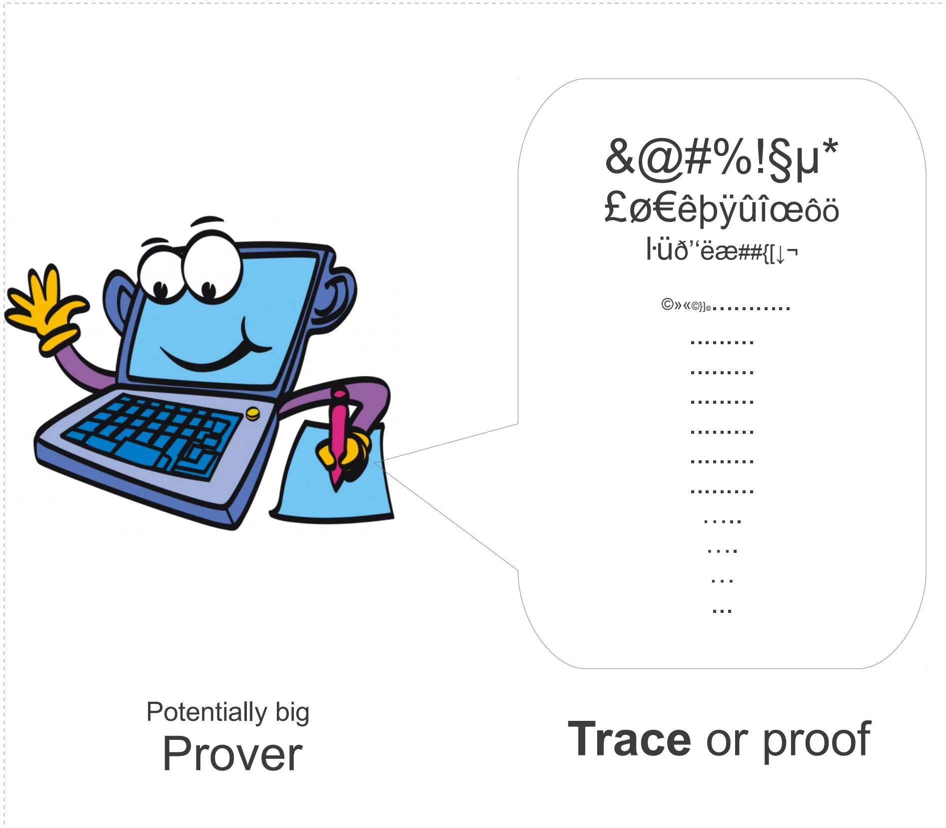
INRIA-Saclay & LIX, Ecole Polytechnique

12 June 2013

Can we standardize, communicate, and trust formal proofs?

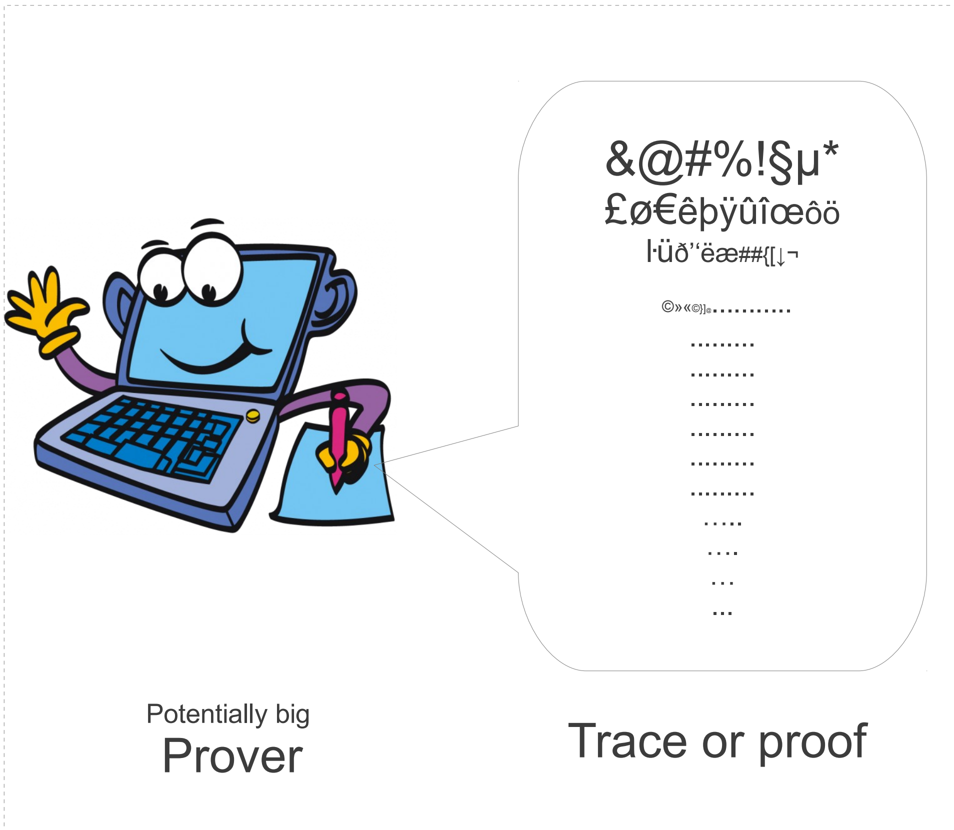
The topic of the ProofCert project

How to trust a machine-generated proof



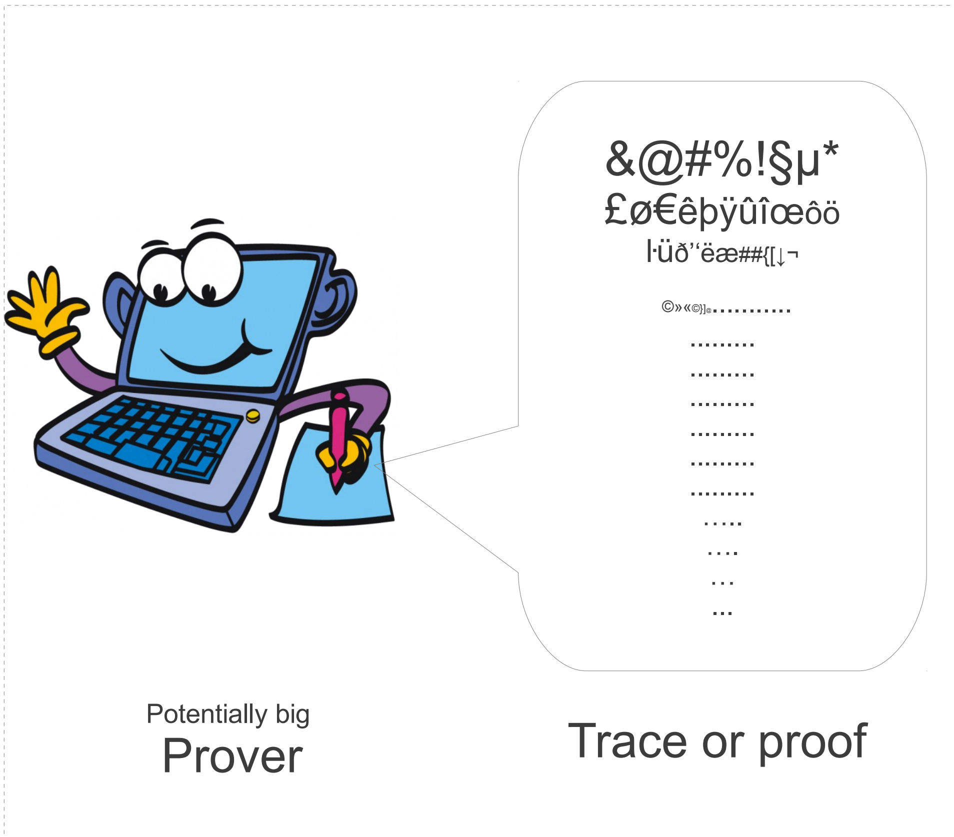
- Read the output or redo the proof
- Trust the prover
  - Formally prove it
  - Build it around a small trusted kernel
- Have a small dedicated checker verify the proof

How to trust a machine-generated proof



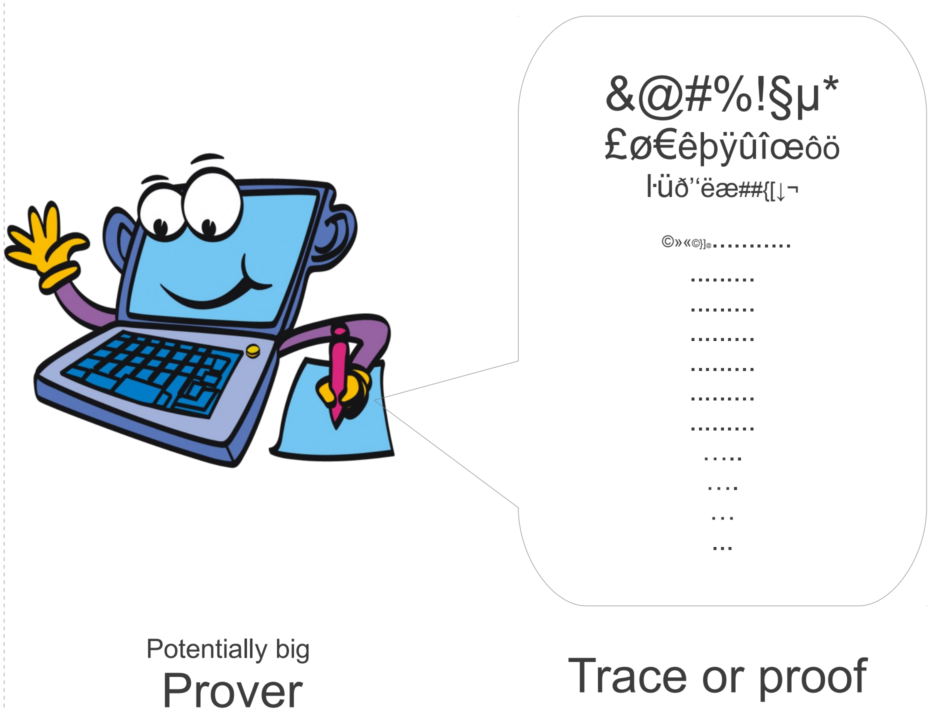
- Read the output or redo the proof
- Trust the prover
  - Formally prove it
  - Build it around a small trusted kernel
- Have a small dedicated checker verify the proof
- How about other provers' proofs?
  - Previous steps
  - Translate their output into your formalism and run them on your prover...

How to trust a machine-generated proof



- Read the output or redo the proof
- Trust the prover
  - Formally prove it
  - Build it around a small trusted kernel
- Have a **small** dedicated checker verify the proof

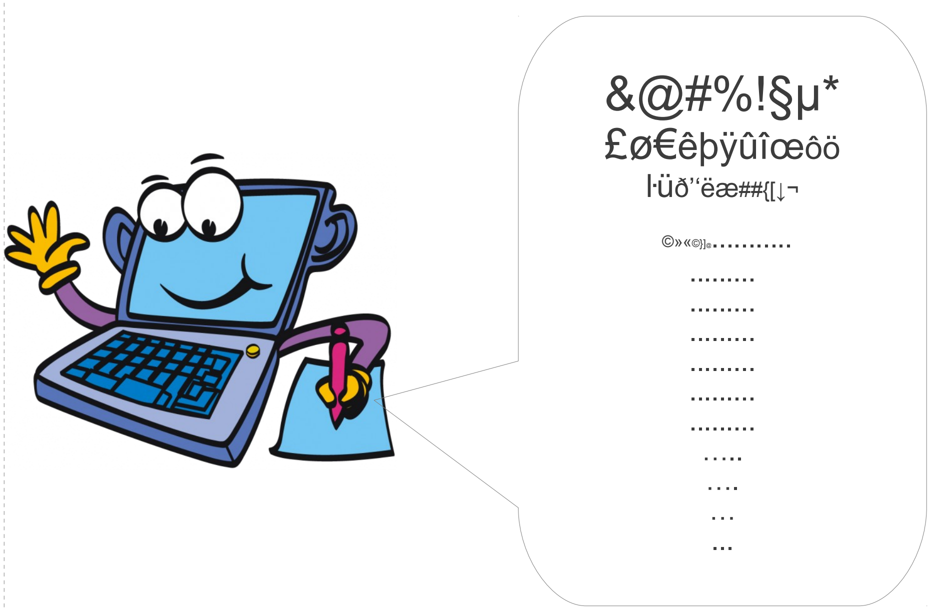
How to trust a machine-generated proof



Human readable

- Have a **small** dedicated checker verify the proof

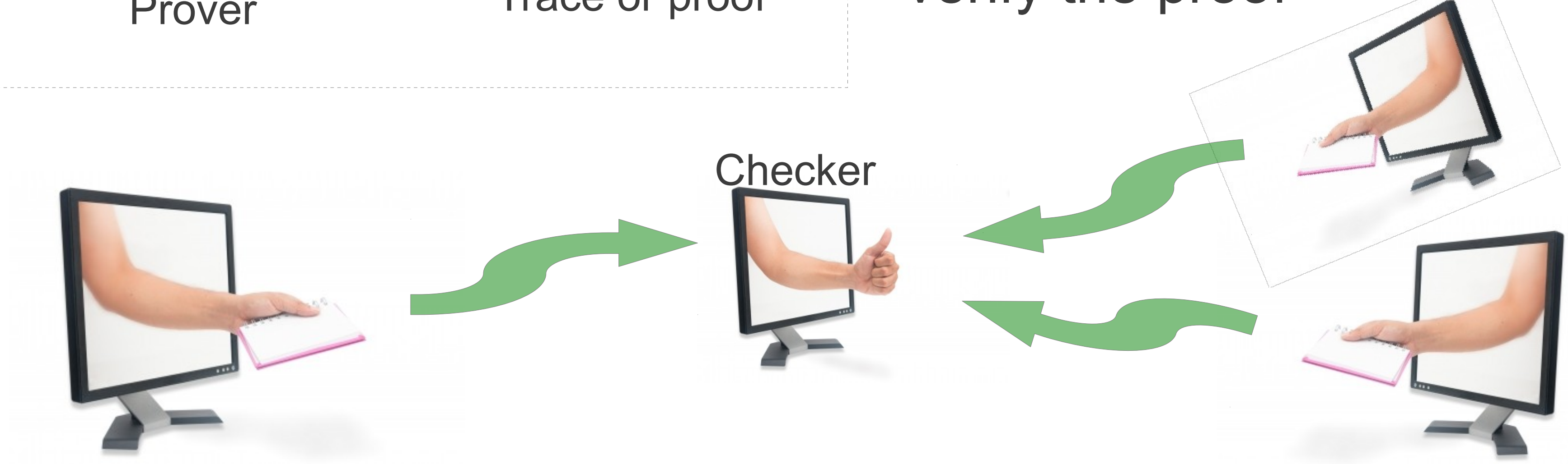
How to trust a machine-generated proof



Potentially big
Prover

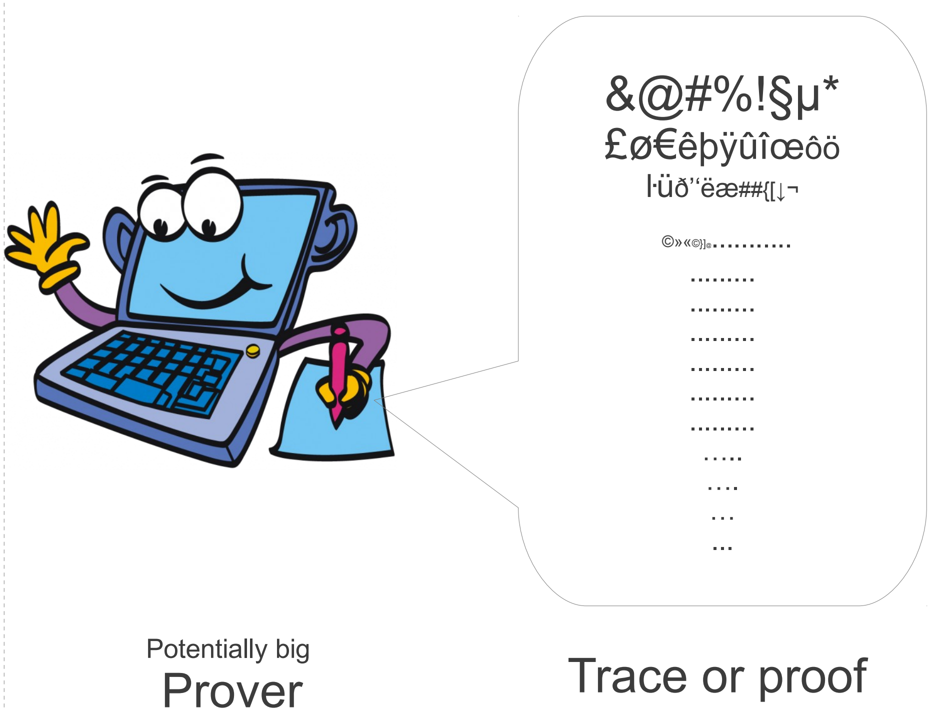
Trace or proof

- Have a small ~~dedicated~~ checker verify the proof

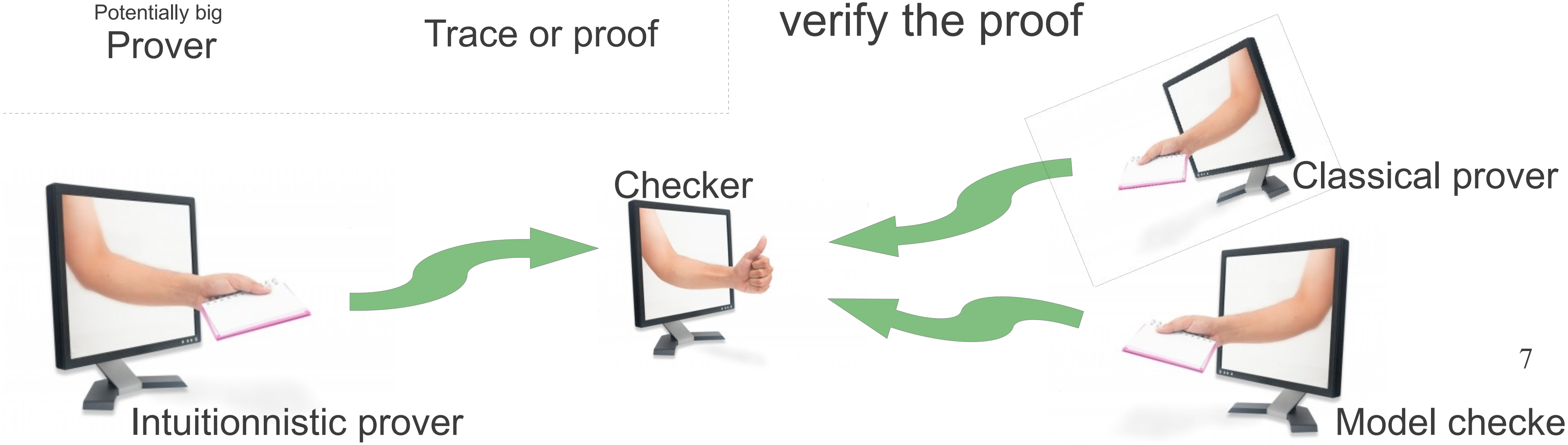


Easily trusted code | **Broad range**

How to trust a machine-generated proof

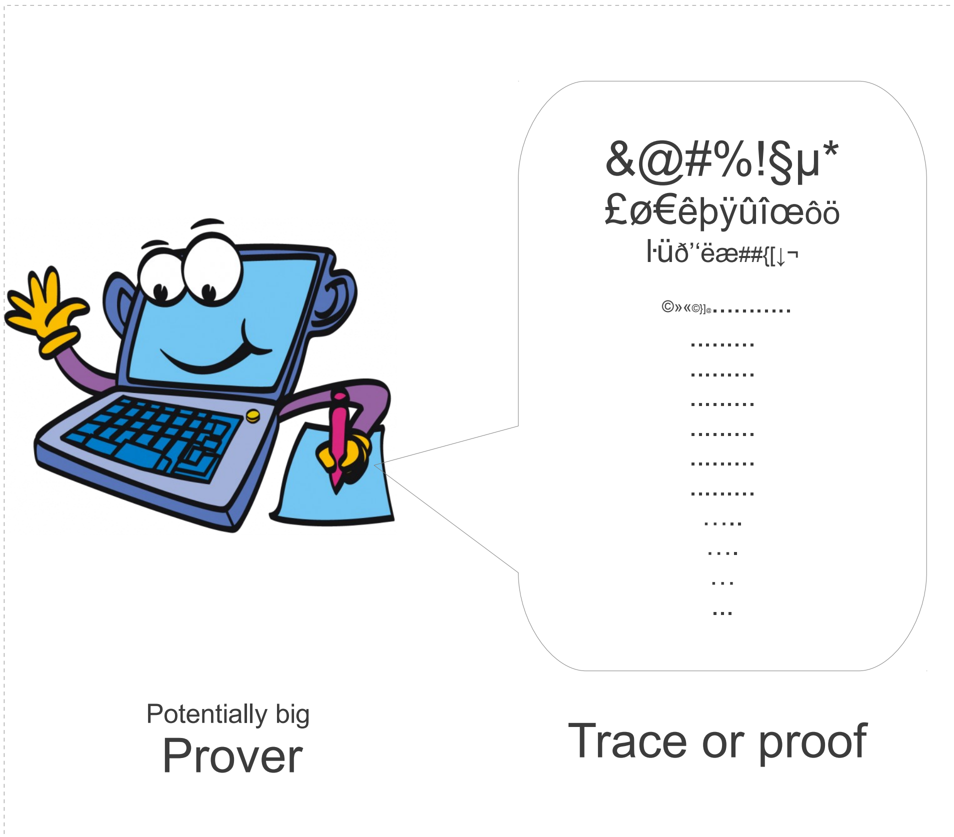


- Have a small **broad-range** checker verify the proof



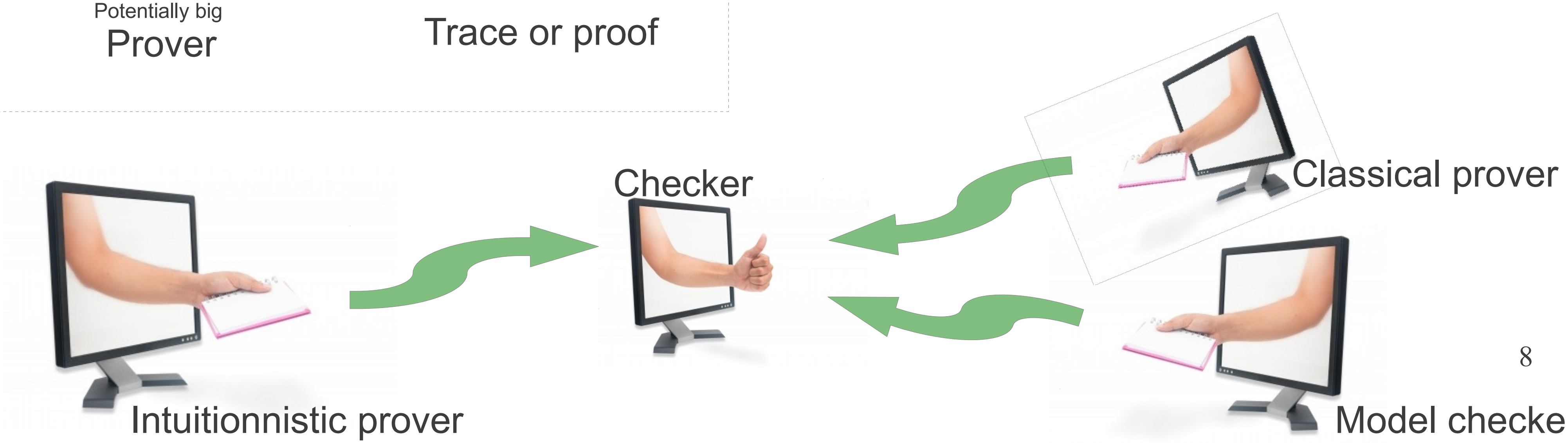
☐ Easily trusted code ☐ Broad range

How to **check** a machine-generated proof



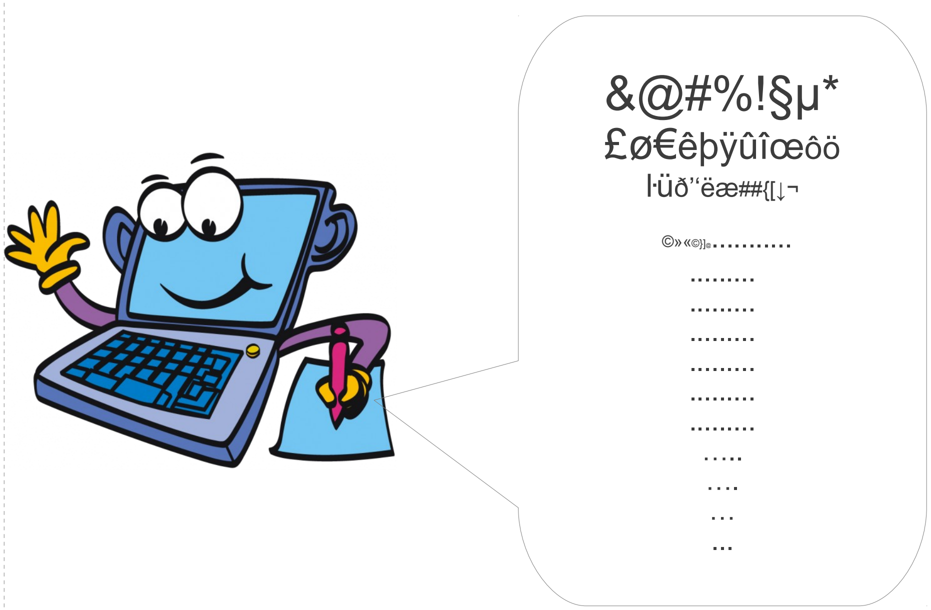
Have a **small broad-range** checker verify the proof

Small **while** « understanding » multiple provers?



☐ Easily trusted code ☐ Broad range

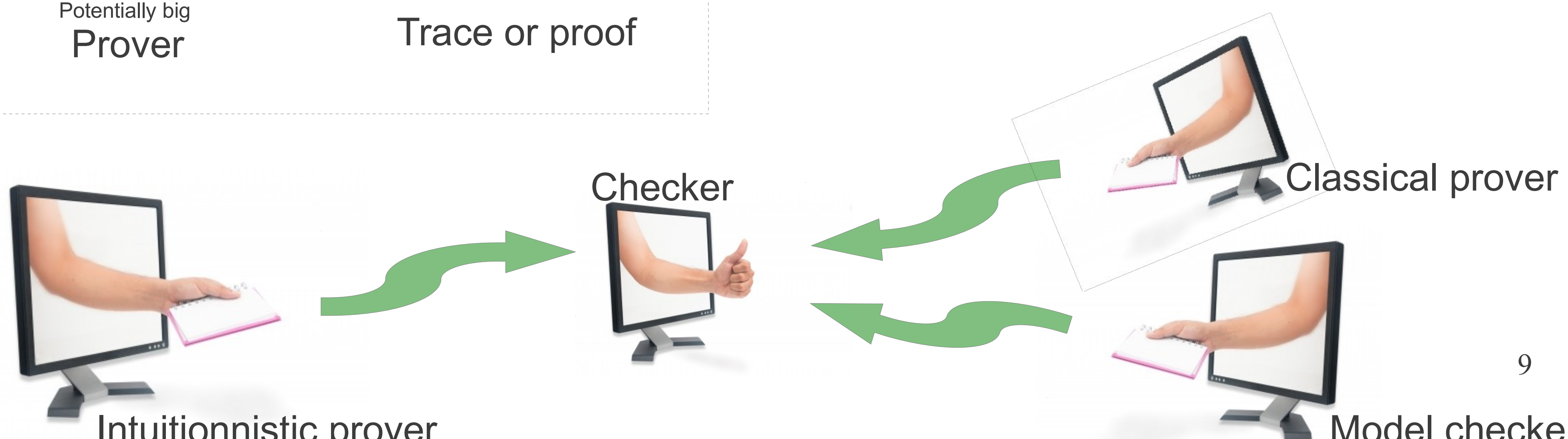
How to **check** a machine-generated proof



Potentially big
Prover

Trace or proof

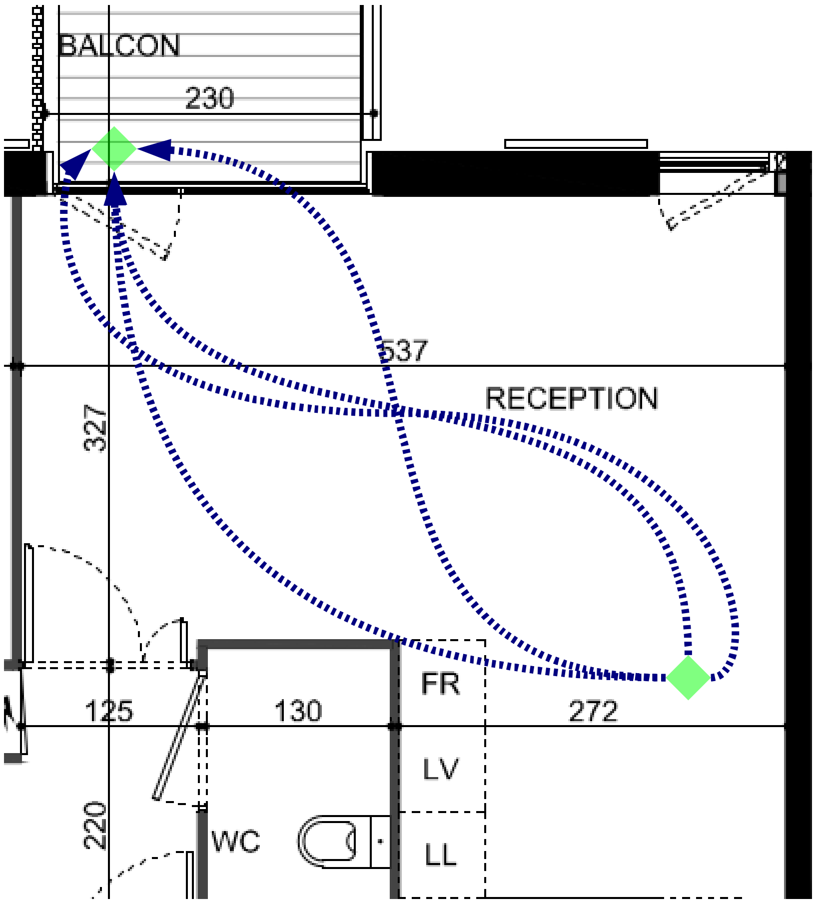
Library of theorems



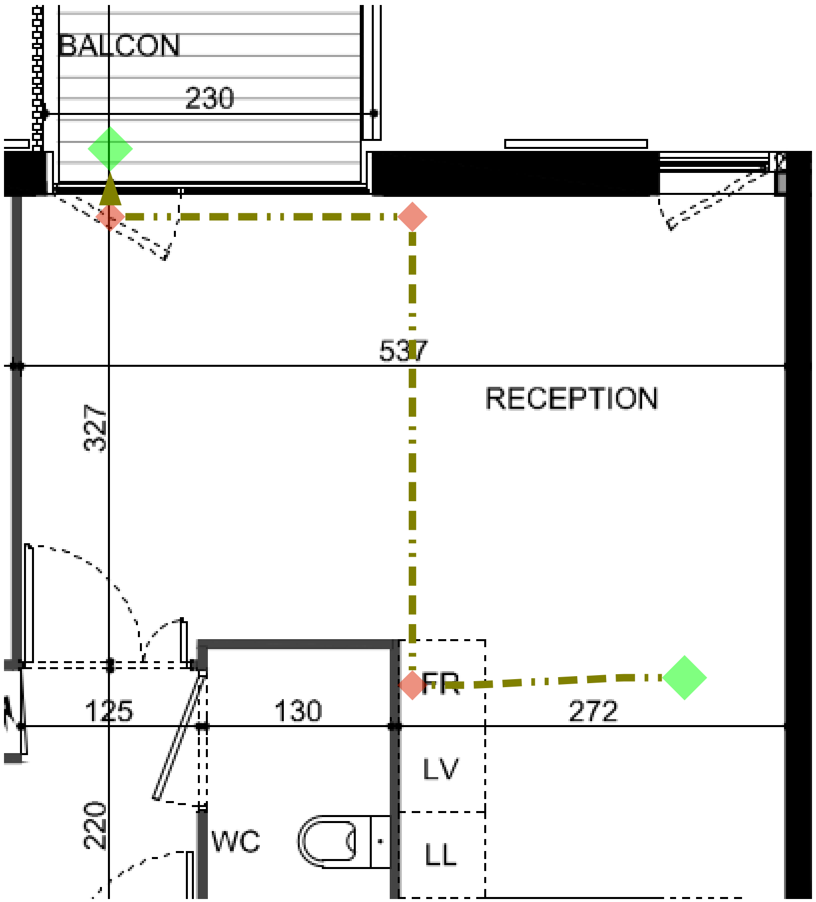
Easily trusted code

Broad range

The kernel of checker: *focused* LK



(Unfocused) sequent calculus



Focused sequent calculus

The kernel of checker: *focused* LK

Focusing $\leftarrow$ Polarities $\leftarrow$ **Invertible**

$$\frac{\vdash \Theta, B_i}{\vdash \Theta, B_1 \vee B_2}\ i \in \{1,2\}$$

Conclusion
⇕
Premise

$$\frac{\vdash \Theta, B_1, B_2}{\vdash \Theta, B_1 \vee B_2}$$

The kernel of checker: *focused* LK

Focusing ← **Polarities** ← Invertible

Simple notations. If you want the connective (or atom) to be subject to

- Invertible rule => give negative polarity
- Non (necessarily) invertible rule => give positive polarity

Chose left!

Yes!
I'm **positive!**

Mnemonics

Are you sure?

$$\frac{\vdash \Theta, B_i}{\vdash \Theta, B_1 \vee^{+} B_2}\ i \in \{1,2\} \qquad\qquad \frac{\vdash \Theta, B_1, B_2}{\vdash \Theta, B_1 \vee^{-} B_2}$$

The kernel of checker: *focused* LK

Focusing ← **Polarities** ← Invertible

Simple **notations**. Connective (or atom) should be subject to

- Invertible rule => negative polarity
- Non (necessarily) invertible rule => positive polarity

Where there is a choice, the checker can be guided. Without leading it to errors?

$$\frac{\vdash \Theta, B_i}{\vdash \Theta, B_1 \vee^{+} B_2}\ i \in \{1,2\} \qquad\qquad \frac{\vdash \Theta, B_1, B_2}{\vdash \Theta, B_1 \vee^{-} B_2}$$

Easily trusted code | Broad range | Sound interaction

The kernel of checker: *focused* LK

Focusing ← Polarities ← Invertible

Organizing proofs in layers of **negative** and **positive (focused)** phases

Negative phase

Sequents :

$$\vdash \Theta \Uparrow \Gamma$$

- Only invertible rules
- No loss of information
- Same input => same output
- Rules applied in any order to negative formulas

Focused or positive phase

Sequents :

More mnemonics

$$\vdash \Theta \Downarrow P$$

- Only non invertible rules
- Selection of information
- Output depends on choices
- Rules applied hereditarily on subformulas of *P*

The kernel of checker: *focused* LK

From the completeness of LKF:

$$\vdash_{LK} A \quad \Leftrightarrow \quad \vdash_{LKF} \; . \Uparrow A^p$$

Where A^p is the a polarized version of A (exponentially many such versions)
e.g. If A = $a \vee b \wedge c$, A^p can be either
$a \vee^- b \wedge^+ c$, $a \vee^- b \wedge^- c$, $a \vee^+ b \wedge^- c$, *etc.*
(The atoms are also polarized)

From now on, $\vdash$ *is taken to be* $\vdash_{LKF}$ *and formulas are considered to be polarized and in negation normal form.*

The kernel of checker: *focused* LK

Negative phase

$$\frac{}{\vdash \Theta \Uparrow t^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, \Gamma \quad \vdash \Theta \Uparrow B, \Gamma}{\vdash \Theta \Uparrow A \wedge^- B, \Gamma} \quad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow f^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, B, \Gamma}{\vdash \Theta \Uparrow A \vee^- B, \Gamma}$$

$$\frac{\vdash \Theta \Uparrow [y/x]B, \Gamma \quad y \text{ not free in } \Theta, \Gamma, B}{\vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Focused or positive phase

$$\frac{}{\vdash \Theta \Downarrow t^+} \quad \frac{\vdash \Theta \Downarrow B_1 \quad \vdash \Theta \Downarrow B_2}{\vdash \Theta \Downarrow B_1 \wedge^+ B_2} \quad \frac{\vdash \Theta \Downarrow B_i \quad i \in \{1,2\}}{\vdash \Theta \Downarrow B_1 \vee^+ B_2} \quad \frac{\vdash \Theta \Downarrow [t/x]B}{\vdash \Theta \Downarrow \exists x.B}$$

The kernel of checker: *focused* LK

Negative phase

$$\frac{}{\vdash \Theta \Uparrow t^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, \Gamma \quad \vdash \Theta \Uparrow B, \Gamma}{\vdash \Theta \Uparrow A \wedge^- B, \Gamma} \quad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow f^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, B, \Gamma}{\vdash \Theta \Uparrow A \vee^- B, \Gamma}$$

$$\frac{\vdash \Theta \Uparrow [y/x]B, \Gamma \quad y \text{ not free in } \Theta, \Gamma, B}{\vdash \Theta \Uparrow \forall x.B, \Gamma}$$

In between

$$\frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow C, \Gamma}\ \mathit{store} \quad \frac{\vdash P, \Theta \Downarrow P}{\vdash P, \Theta \Uparrow \cdot}\ \mathit{decide} \quad \frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\ \mathit{release} \quad \frac{}{\vdash \neg P_a, \Theta \Downarrow P_a}\ \mathit{init}$$

Focused or positive phase

$$\frac{}{\vdash \Theta \Downarrow t^+} \quad \frac{\vdash \Theta \Downarrow B_1 \quad \vdash \Theta \Downarrow B_2}{\vdash \Theta \Downarrow B_1 \wedge^+ B_2} \quad \frac{\vdash \Theta \Downarrow B_i \quad i \in \{1, 2\}}{\vdash \Theta \Downarrow B_1 \vee^+ B_2} \quad \frac{\vdash \Theta \Downarrow [t/x]B}{\vdash \Theta \Downarrow \exists x.B}$$

Easily trusted code | Broad range | Sound interaction

The kernel of checker: *focused* LK

Negative phase

$$\frac{}{\vdash \Theta \Uparrow t^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, \Gamma \quad \vdash \Theta \Uparrow B, \Gamma}{\vdash \Theta \Uparrow A \wedge^- B, \Gamma} \quad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow f^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, B, \Gamma}{\vdash \Theta \Uparrow A \vee^- B, \Gamma}$$

$$\frac{\vdash \Theta \Uparrow [y/x]B, \Gamma \quad y \text{ not free in } \Theta, \Gamma, B}{\vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Only contract on positive

In between

$$\frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow C, \Gamma}\ store \qquad \frac{\vdash P, \Theta \Downarrow P}{\vdash P, \Theta \Uparrow \cdot}\ decide \qquad \frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\ release \qquad \frac{}{\vdash \neg P_a, \Theta \Downarrow P_a}\ init$$

Focused or positive phase

$$\frac{}{\vdash \Theta \Downarrow t^+} \quad \frac{\vdash \Theta \Downarrow B_1 \quad \vdash \Theta \Downarrow B_2}{\vdash \Theta \Downarrow B_1 \wedge^+ B_2} \quad \frac{\vdash \Theta \Downarrow B_i \quad i \in \{1,2\}}{\vdash \Theta \Downarrow B_1 \vee^+ B_2} \quad \frac{\vdash \Theta \Downarrow [t/x]B}{\vdash \Theta \Downarrow \exists x.B}$$

The kernel of checker: *focused* LK

Negative phase

$$\frac{}{\vdash \Theta \Uparrow t^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, \Gamma \quad \vdash \Theta \Uparrow B, \Gamma}{\vdash \Theta \Uparrow A \wedge^- B, \Gamma} \quad \frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow f^-, \Gamma} \quad \frac{\vdash \Theta \Uparrow A, B, \Gamma}{\vdash \Theta \Uparrow A \vee^- B, \Gamma}$$

$$\frac{\vdash \Theta \Uparrow [y/x]B, \Gamma \quad y \text{ not free in } \Theta, \Gamma, B}{\vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Only contract on positive

In between

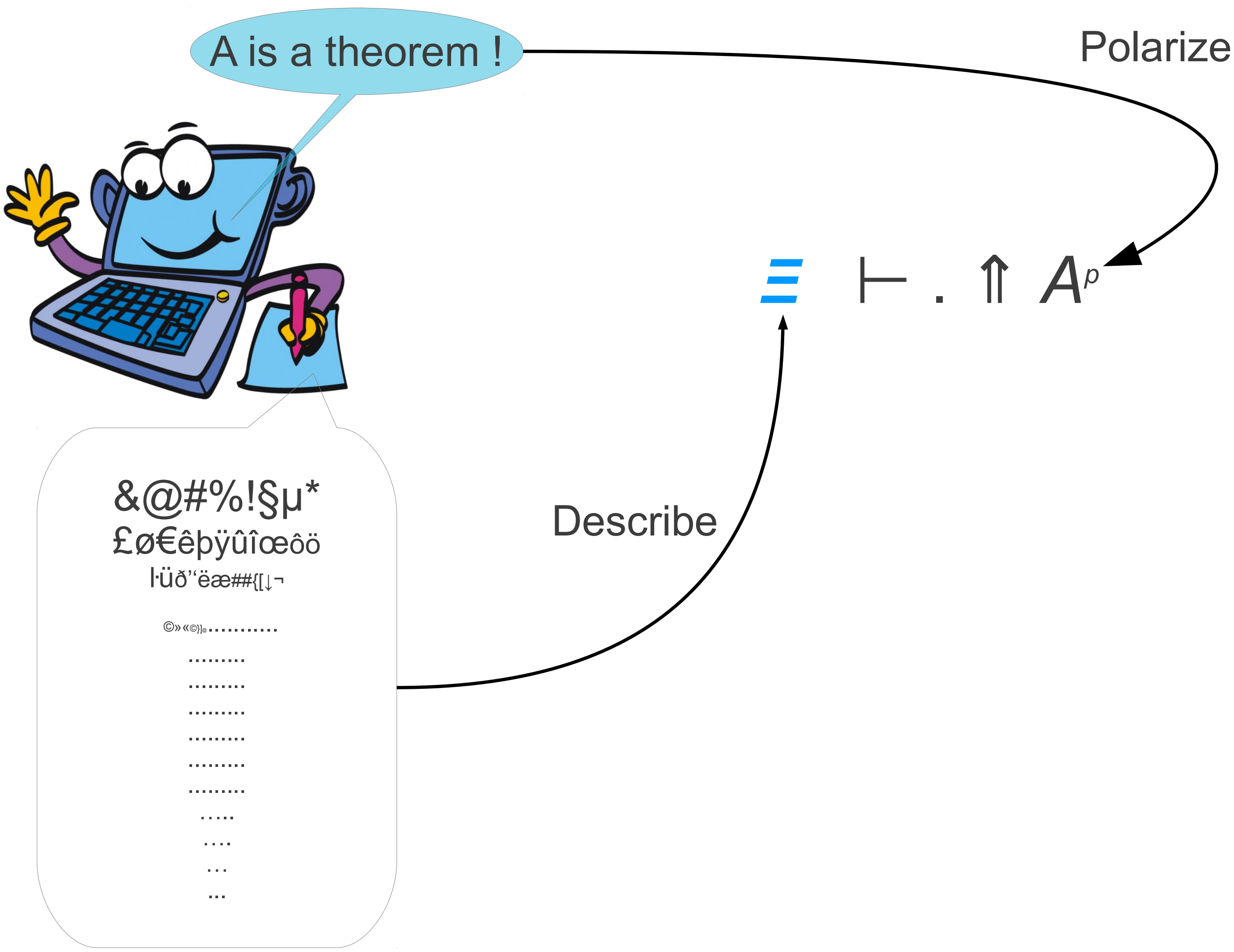
$$\frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow C, \Gamma}\ store \qquad \frac{\vdash P, \Theta \Downarrow P}{\vdash P, \Theta \Uparrow \cdot}\ decide \qquad \frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\ release \qquad \frac{}{\vdash \neg P_a, \Theta \Downarrow P_a}\ init$$

Focused or positive phase

$$\frac{}{\vdash \Theta \Downarrow t^+} \quad \frac{\vdash \Theta \Downarrow B_1 \quad \vdash \Theta \Downarrow B_2}{\vdash \Theta \Downarrow B_1 \wedge^+ B_2} \quad \frac{\vdash \Theta \Downarrow B_i \quad i \in \{1,2\}}{\vdash \Theta \Downarrow B_1 \vee^+ B_2} \quad \frac{\vdash \Theta \Downarrow [t/x]B}{\vdash \Theta \Downarrow \exists x.B}$$

☐ Easily trusted code ☐ Broad range ☐ Sound interaction

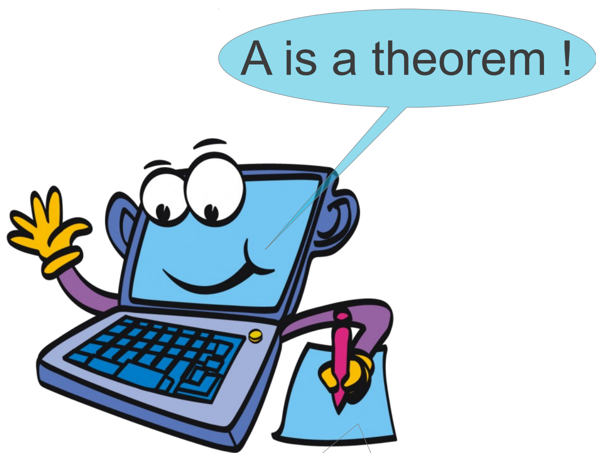
Back to checking, LKF[a] *(augmented* LKF)



Trace or proof

Easily trusted code ☐ Broad range ☐ Sound interaction ☐

Back to checking, LKF[a] *(augmented* LKF)



Polarize

$$\Xi \;\vdash\; . \Uparrow A^p$$

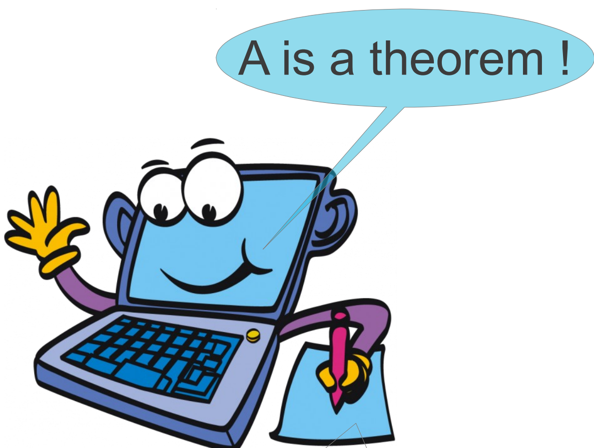
But how to **feed** information, when needed, to the kernel?

Describe

&@#%!§µ*
£ø€êþÿûîœôö
l·üð’‘ëæ##{[↓¬
©»«©}|...........
....
....
....
....
....
....
.....
....
...
...

Trace or proof

Back to checking, LKF^a *(augmented* LKF)



Polarize

$$\Xi \ \vdash \ . \Uparrow A^p$$

But how to **feed** information, when needed, to the kernel?

Describe

&@#%!§µ*
£ø€êþÿûîœôö
l·üð'‘ëæ##{[↓¬
©»«©}|ₒ..........
....
....
....
....
....
....
.....
....
...
...

Trace or proof

$$\frac{\vdash \Theta \Downarrow B_i \qquad i \in \{1,2\}}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2}$$

Easily trusted code | Broad range | Sound interaction

Back to checking, LKF[a] *(augmented* LKF)

A is a theorem !

Polarize

$$\Xi \vdash . \Uparrow A^p$$

But how to **feed** information, when needed, to the kernel?

&@#%!§µ*
£ø€êþÿûîœôö
ł·üð'‘ëæ##{[↓¬
©»«©}}ₒ..........
....
....
....
....
....
....
.....
....
...
...

Describe

Trace or proof

Expert

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1,2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2}$$

Chose left ; Ξ'

Easily trusted code | Broad range | Sound interaction

Back to checking, LKF[a] *(augmented* LKF)

Polarize

$$\Xi \vdash . \Uparrow A^p$$

But how to **feed** information, when needed, to the kernel?
What if the information is **not** there?

&@#%!§µ*
£ø€êþÿûîœôö
l·üð''ëæ##{[↓¬
©»«©}}ₒ..........
....
....
....
....
....
....
.....
....
...
...

Describe

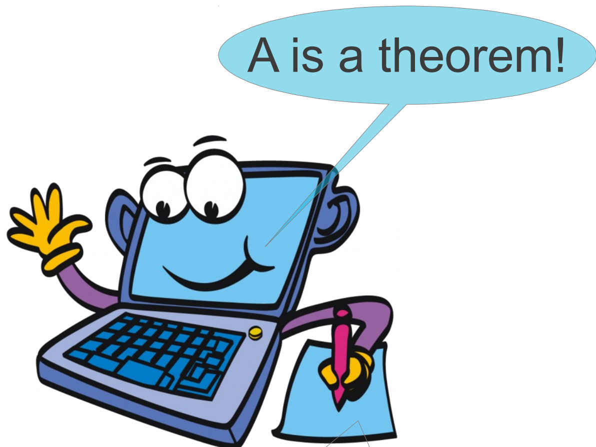
Expert

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1,2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2}$$

Chose left...or...or right...Definitely one of these two... Ξ'

Trace or proof

Back to checking, LKF[a] *(augmented* LKF)



Polarize

$$\Xi \vdash . \Uparrow A^p$$

Describe

&@#%!§µ*
£ø€êþÿûîœôö
ł·üð'‘ëæ##{[↓¬
©»«©}]ₒ..........
....
....
....
....
....
....
.....
....
...
...

Trace or proof

But how to **feed** information, when needed, to the kernel?
What if the information is **not** there?

Expert

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1,2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2}$$

Chose left...or...or right...Definitely one of these two... Ξ'

Positive phase

- And we do the same each time we may **guide** the proof checking!

$$\frac{t_e(\Xi)}{\Xi \vdash \Theta \Downarrow t^+} \qquad \frac{\Xi_1 \vdash \Theta \Downarrow B_1 \qquad \Xi_2 \vdash \Theta \Downarrow B_2 \qquad \wedge_e(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1,2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2} \qquad \frac{\Xi' \vdash \Theta \Downarrow [t/x]B \qquad \exists_e(\Xi, \Xi', t)}{\Xi \vdash \Theta \Downarrow \exists x.B}$$

Positive phase

- And we do the same each time we may **guide** the proof checking!

$$\frac{t_e(\Xi)}{\Xi \vdash \Theta \Downarrow t^+} \qquad \frac{\Xi_1 \vdash \Theta \Downarrow B_1 \qquad \Xi_2 \vdash \Theta \Downarrow B_2 \qquad \wedge_e(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1,2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2} \qquad \frac{\Xi' \vdash \Theta \Downarrow [t/x]B \qquad \exists_e(\Xi, \Xi', t)}{\Xi \vdash \Theta \Downarrow \exists x.B}$$

- The witness is *t*!
- The witness *t* is in the set *S*, but I don't know which...
- The witness is … wait, what witness?

Positive phase

- And we do the same each time we may **guide** the proof checking!

$$\frac{t_e(\Xi)}{\Xi \vdash \Theta \Downarrow t^+} \qquad \frac{\Xi_1 \vdash \Theta \Downarrow B_1 \qquad \Xi_2 \vdash \Theta \Downarrow B_2 \qquad \wedge_e(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1, 2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2} \qquad \frac{\Xi' \vdash \Theta \Downarrow [t/x]B \qquad \exists_e(\Xi, \Xi', t)}{\Xi \vdash \Theta \Downarrow \exists x.B}$$

Backtrack!

Unification!

Positive phase

- And we do the same each time we may **guide** the proof checking!

$$\frac{t_e(\Xi)}{\Xi \vdash \Theta \Downarrow t^+} \qquad \frac{\Xi_1 \vdash \Theta \Downarrow B_1 \qquad \Xi_2 \vdash \Theta \Downarrow B_2 \qquad \wedge_e(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \qquad i \in \{1, 2\} \qquad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2} \qquad \frac{\Xi' \vdash \Theta \Downarrow [t/x]B \qquad \exists_e(\Xi, \Xi', t)}{\Xi \vdash \Theta \Downarrow \exists x.B}$$

Let's give him the wrong witness!

☐ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ☐ Sound interaction

Negative phase

- Negative phase needs no steering. Simple bookkeeping :

It went left

It went right

$$\frac{\Xi' \vdash \Theta \Uparrow \Gamma \quad f_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow f^-, \Gamma} \qquad \frac{\Xi_1 \vdash \Theta \Uparrow A, \Gamma \quad \Xi_2 \vdash \Theta \Uparrow B, \Gamma \quad \wedge_c(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \Uparrow A, B, \Gamma \quad \vee_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow A \vee^- B, \Gamma} \qquad \frac{\Xi' \vdash \Theta \Uparrow [y/x]B, \Gamma \quad \forall_c(\Xi, \Xi') \quad y \text{ not free in } \Xi, \Theta, \Gamma, B}{\Xi \vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Clerk

☐ Easily trusted code ☐ Broad range ☑ Flexible reconstruction ☐ Sound interaction

Negative phase

- Negative phase needs no steering. Simple bookkeeping :

Part relative to the left branch

Part relative to the right branch

$$\frac{\Xi' \vdash \Theta \Uparrow \Gamma \quad f_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow f^-, \Gamma} \qquad \frac{\Xi_1 \vdash \Theta \Uparrow A, \Gamma \quad \Xi_2 \vdash \Theta \Uparrow B, \Gamma \quad \wedge_c(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \Uparrow A, B, \Gamma \quad \vee_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow A \vee^- B, \Gamma} \qquad \frac{\Xi' \vdash \Theta \Uparrow [y/x]B, \Gamma \quad \forall_c(\Xi, \Xi') \quad y \text{ not free in } \Xi, \Theta, \Gamma, B}{\Xi \vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Clerk

☐ Easily trusted code ☐ Broad range ☑ Flexible reconstruction ☐ Sound interaction

Negative phase

- Negative phase needs no steering. Simple bookkeeping :

No work done

No work done

$$\frac{\Xi' \vdash \Theta \Uparrow \Gamma \quad f_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow f^-, \Gamma} \qquad \frac{\Xi \ \vdash \Theta \Uparrow A, \Gamma \quad \Xi \ \vdash \Theta \Uparrow B, \Gamma \quad \wedge_c(\Xi, \Xi \ , \Xi \ )}{\Xi \vdash \Theta \Uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \Uparrow A, B, \Gamma \quad \vee_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow A \vee^- B, \Gamma} \qquad \frac{\Xi' \vdash \Theta \Uparrow [y/x]B, \Gamma \quad \forall_c(\Xi, \Xi') \quad y \text{ not free in } \Xi, \Theta, \Gamma, B}{\Xi \vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Clerk

☐ Easily trusted code ☐ Broad range ☑ Flexible reconstruction ☐ Sound interaction

Negative phase

- Negative phase needs no steering. Simple bookkeeping :

$$\frac{\Xi' \vdash \Theta \Uparrow \Gamma \quad f_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow f^-, \Gamma} \qquad \frac{\Xi \vdash \Theta \Uparrow A, \Gamma \quad \Xi \vdash \Theta \Uparrow B, \Gamma \quad \wedge_c(\Xi, \Xi, \Xi)}{\Xi \vdash \Theta \Uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \Uparrow A, B, \Gamma \quad \vee_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow A \vee^- B, \Gamma} \qquad \frac{\Xi' \vdash \Theta \Uparrow [y/x]B, \Gamma \quad \forall_c(\Xi, \Xi') \quad y \text{ not free in } \Xi, \Theta, \Gamma, B}{\Xi \vdash \Theta \Uparrow \forall x.B, \Gamma}$$

Succeed on any input

☐ Easily trusted code ☐ Broad range ☑ Flexible reconstruction ☐ Sound interaction

$$\frac{t_e(\Xi)}{\Xi \vdash \Theta \Downarrow t^+} \qquad \frac{\Xi_1 \vdash \Theta \Downarrow B_1 \quad \Xi_2 \vdash \Theta \Downarrow B_2 \quad \wedge_e(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Downarrow B_1 \wedge^+ B_2}$$

$$\frac{\Xi' \vdash \Theta \Downarrow B_i \quad i \in \{1,2\} \quad \vee_e(\Xi, \Xi', i)}{\Xi \vdash \Theta \Downarrow B_1 \vee^+ B_2} \qquad \frac{\Xi' \vdash \Theta \Downarrow [t/x]B \quad \exists_e(\Xi, \Xi', t)}{\Xi \vdash \Theta \Downarrow \exists x.B}$$

$$\frac{\Xi_1 \vdash \Theta \Uparrow B \quad \Xi_2 \vdash \Theta \Uparrow \neg B \quad cut_e(\Xi, \Theta, \Xi_1, \Xi_2, B)}{\Xi \vdash \Theta \Uparrow \cdot}\ cut$$

$$\frac{\Xi' \vdash \Theta \Uparrow N \quad release_e(\Xi, \Xi')}{\Xi \vdash \Theta \Downarrow N}\ release \qquad \frac{init_e(\Xi, \Theta, l) \quad \langle l, \neg P_a \rangle \in \Theta}{\Xi \vdash \Theta \Downarrow P_a}\ init$$

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot}\ decide$$

$$\frac{\Xi' \vdash \Theta \Uparrow \Gamma \quad f_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow f^-, \Gamma} \qquad \frac{\Xi_1 \vdash \Theta \Uparrow A, \Gamma \quad \Xi_2 \vdash \Theta \Uparrow B, \Gamma \quad \wedge_c(\Xi, \Xi_1, \Xi_2)}{\Xi \vdash \Theta \Uparrow A \wedge^- B, \Gamma}$$

$$\frac{\Xi' \vdash \Theta \Uparrow A, B, \Gamma \quad \vee_c(\Xi, \Xi')}{\Xi \vdash \Theta \Uparrow A \vee^- B, \Gamma} \qquad \frac{\Xi' \vdash \Theta \Uparrow [y/x]B, \Gamma \quad \forall_c(\Xi, \Xi') \quad y \text{ not free in } \Xi, \Theta, \Gamma, B}{\Xi \vdash \Theta \Uparrow \forall x.B, \Gamma}$$

$$\frac{}{\Xi \vdash \Theta \Uparrow t^-, \Gamma} \qquad \frac{\Xi' \vdash \Theta, \langle l, C \rangle \Uparrow \Gamma \quad store_c(\Xi, C, \Xi', l)}{\Xi \vdash \Theta \Uparrow C, \Gamma}\ store$$

Here, P is a positive formula; N a negative formula; P_a a positive literal; C a positive formula or negative literal. In the cut rule, the expression $\neg B$ is the negation of B (defined on connectives as the usual first-order classical negation with polarity flip, on literals as a single polarity flip).

☐ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

$$\frac{}{\vdash \Theta \Downarrow t^{+}} \qquad \frac{\vdash \Theta \Downarrow B_1 \qquad \vdash \Theta \Downarrow B_2}{\vdash \Theta \Downarrow B_1 \wedge^{+} B_2}$$

$$\frac{\vdash \Theta \Downarrow B_i \qquad i \in \{1,2\}}{\vdash \Theta \Downarrow B_1 \vee^{+} B_2} \qquad \frac{\vdash \Theta \Downarrow [t/x]B}{\vdash \Theta \Downarrow \exists x.B}$$

$$\frac{\vdash \Theta \Uparrow B \qquad \vdash \Theta \Uparrow \neg B}{\vdash \Theta \Uparrow \cdot}\ cut$$

$$\frac{\vdash \Theta \Uparrow N}{\vdash \Theta \Downarrow N}\ release \qquad \frac{\neg P_a \in \Theta}{\vdash \Theta \Downarrow P_a}\ init$$

$$\frac{\vdash \Theta \Downarrow P \qquad P \in \Theta \quad \text{positive}(P)}{\vdash \Theta \Uparrow \cdot}\ decide$$

$$\frac{\vdash \Theta \Uparrow \Gamma}{\vdash \Theta \Uparrow f^{-}, \Gamma} \qquad \frac{\vdash \Theta \Uparrow A, \Gamma \qquad \vdash \Theta \Uparrow B, \Gamma}{\vdash \Theta \Uparrow A \wedge^{-} B, \Gamma}$$

$$\frac{\vdash \Theta \Uparrow A, B, \Gamma}{\vdash \Theta \Uparrow A \vee^{-} B, \Gamma} \qquad \frac{\vdash \Theta \Uparrow [y/x]B, \Gamma \qquad y \text{ not free in } \Theta, \Gamma, B}{\vdash \Theta \Uparrow \forall x.B, \Gamma}$$

$$\frac{}{\vdash \Theta \Uparrow t^{-}, \Gamma} \qquad \frac{\vdash \Theta, C \Uparrow \Gamma}{\vdash \Theta \Uparrow C, \Gamma}\ store$$

Here, P is a positive formula; N a negative formula; P_a a positive literal; C a positive formula or negative literal. In the cut rule, the expression $\neg B$ is the negation of B (defined on connectives as the usual first-order classical negation with polarity flip, on literals as a single polarity flip).

✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot} \; decide$$

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\, async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P\rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P\rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot}\ decide$$

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\ async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P\rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \mathrm{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot}\ decide$$

n-1

n

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\ async(\Xi, \Theta, [\,]) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P\rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

Decide on anything but P

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot} \; decide$$

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\, async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\; memb(\langle l, P\rangle, \Theta),\; pos(P),\; sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

Readline

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot}\ decide$$

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\ async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P \rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

Read from the pointer

$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \mathrm{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot}\ decide$$

Pointer to a file

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\ async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P\rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Coding the kernel

- Every rule is **a** Horn clause in λProlog, for example, decide rule:

Call another program

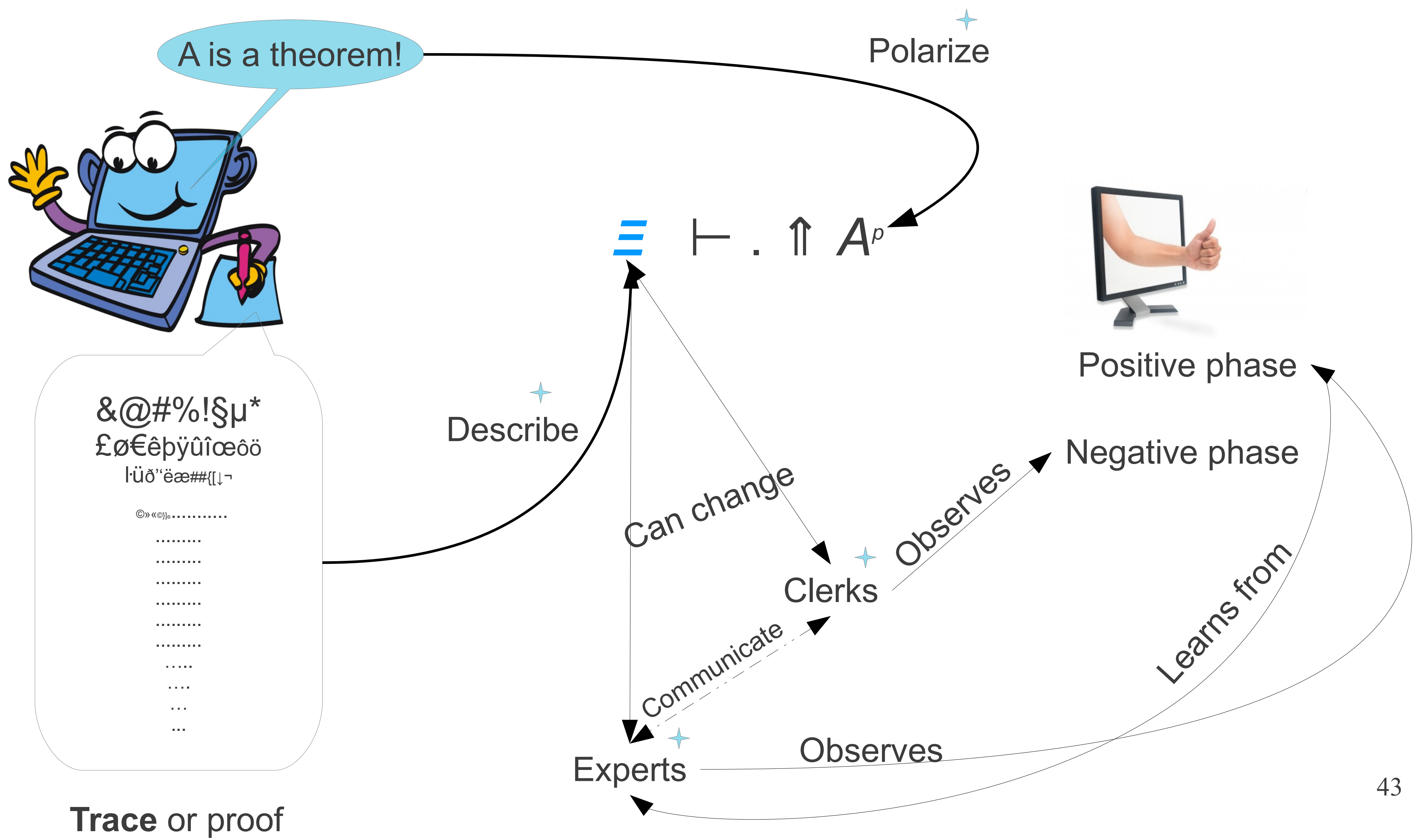
$$\frac{\Xi' \vdash \Theta \Downarrow P \quad decide_e(\Xi, \Theta, \Xi', l) \quad \langle l, P \rangle \in \Theta \quad \text{positive}(P)}{\Xi \vdash \Theta \Uparrow \cdot} \; decide$$

$$\forall\Theta\forall\Xi\forall\Xi'\forall P\forall l.\, async(\Xi, \Theta, []) :- decide_e(\Xi, \Theta, \Xi', l),\ memb(\langle l, P\rangle, \Theta),\ pos(P),\ sync(\Xi', \Theta, P).$$

If P is given, it is checked. If it is not given, *member* will unify with a positive formula in the context: limited backtrack will get to the one that works.

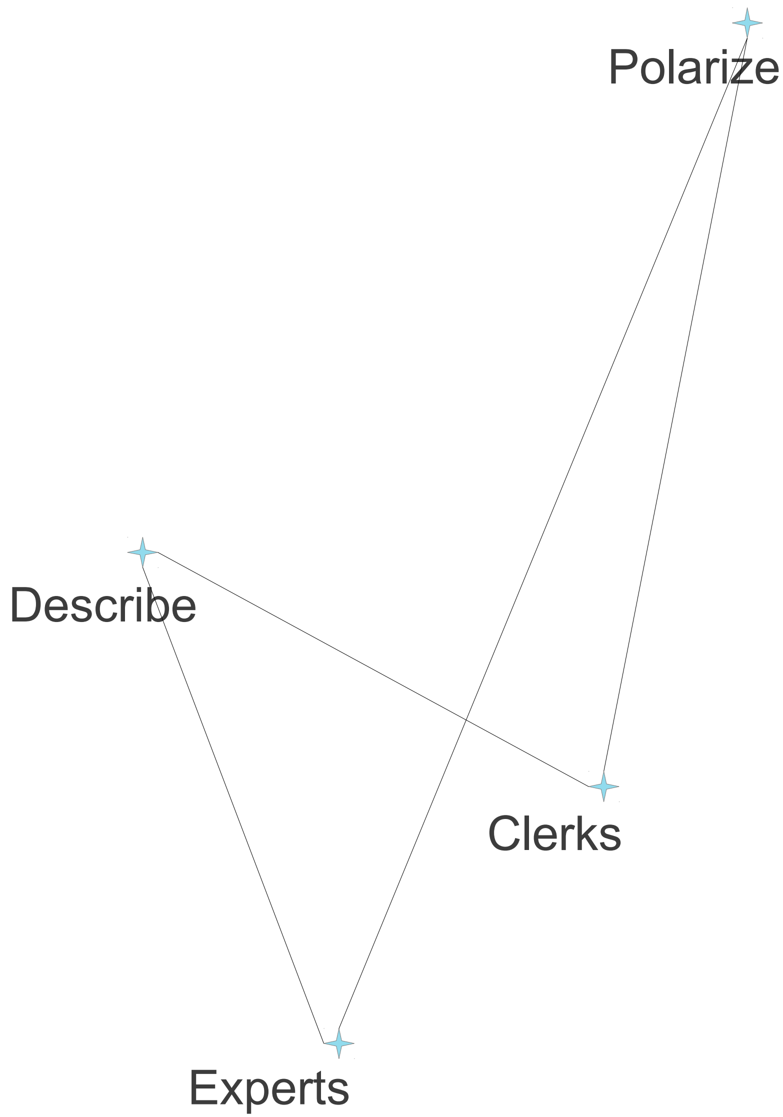
✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Interaction summary



✓ Easily trusted code ☐ Broad range ✓ Flexible reconstruction ✓ Sound interaction

Certificate « constellation »



✓ Easily trusted code ? Broad range ✓ Flexible reconstruction ✓ Sound interaction

The (current) actual kernel

- LKU is a framework of which LKF, LJF and MALLF are subsets.
- Can describe resolution refutation, mating, dependently typed lambda calculus, expansion trees, rewriting …
- Ongoing work for LFSC, LF-modulo, tabled proofs …
- Delighted to work with you!

Future and related work

- Future work
  - Fixpoints, model checkers, improving performance
  - Counter-examples and partial proofs
  - Better formalization of the LKU framework
- Related work
  - Logosphere and OpenTheory
  - TPTP
  - Dedukti