Some Ideas on How to Find Better Proof Representations

Alessio Guglielmi

University of Bath

26 November 2010

This talk is available at `http://cs.bath.ac.uk/ag/t/Hilb24.pdf`.
It requires Acrobat 9 or later.

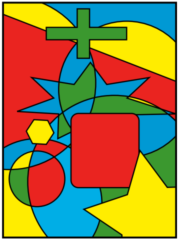
The Dream

The Dream

- No syntax, no symbols, no words.
- An alien could understand this proof.
- Is something like this possible for every proof?

The Reality

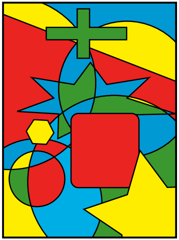
```
Lemma sumt_ctree_pick_rev : forall t t', sumt (ctree_pick_rev t t') = Color0.
Proof.
move=> t' t; rewrite /ctree_pick_rev; set cs0 : colseq := seq0.
have: Color0 +c sumt cs0 = Color0 by done.
elim: t cs0 {1 3}Color0 => [t1 Ht1 t2 Ht2 t3 Ht3|lf _|] et e //.
  move=> Het /=; set cprr := ctree_pick_rev_rec.
  case Det1: (cprr _ _ _ t1) => [|e1 et1].
    case Det2: (cprr _ _ _ t2) => [|e2 et2].
      by apply: Ht3; rewrite [Color3]lock /= -addcA addc_inv.
    by rewrite -Det2; apply: Ht2; rewrite [Color2]lock /= -addcA addc_inv.
  by rewrite -Det1; apply: Ht1; rewrite [Color1]lock /= -addcA addc_inv.
by move=> Het /=; case (ctree_mem t' (etrace (belast e et))).
Qed.
```



- 100s of similar pieces in the four colour theorem proof in Coq.
- Syntactic object with a lot of arbitrary choice: bureaucracy.

The Reality

```
Lemma sumt_ctree_pick_rev : forall t t', sumt (ctree_pick_rev t t') = Color0.
Proof.
move=> t' t; rewrite /ctree_pick_rev; set cs0 : colseq := seq0.
have: Color0 +c sumt cs0 = Color0 by done.
elim: t cs0 {1 3}Color0 => [t1 Ht1 t2 Ht2 t3 Ht3|lf _|] et e //.
  move=> Het /=; set cprr := ctree_pick_rev_rec.
  case Det1: (cprr _ _ _ t1) => [|e1 et1].
    case Det2: (cprr _ _ _ t2) => [|e2 et2].
      by apply: Ht3; rewrite [Color3]lock /= -addcA addc_inv.
    by rewrite -Det2; apply: Ht2; rewrite [Color2]lock /= -addcA addc_inv.
  by rewrite -Det1; apply: Ht1; rewrite [Color1]lock /= -addcA addc_inv.
by move=> Het /=; case (ctree_mem t' (etrace (belast e et))).
Qed.
```



- 100s of similar pieces in the four colour theorem proof in Coq.
- Syntactic object with a lot of arbitrary choice: bureaucracy.

Questions:

- How do we determine whether two proofs are 'the same'?
- Can we free proofs from the idiosyncrasies of language?

Strategy:
We conserve the existing proof theory properties ...

Gentzen's major breakthrough (1930s):

- proofs can be analytic, *i.e.*, built in finitary ways,
- by horribly expensive algorithms,
- that nonetheless allow us to control and analyse them.

Strategy:
We conserve the existing proof theory properties . . .

Gentzen's major breakthrough (1930s):

- proofs can be analytic, *i.e.*, built in finitary ways,
- by horribly expensive algorithms,
- that nonetheless allow us to control and analyse them.

$$
{\scriptstyle \supset_R}\dfrac{{\scriptstyle \vee_L}\dfrac{{\scriptstyle \supset_L}\dfrac{{\scriptstyle \vee_{RL}}\dfrac{a \vdash a}{a \vdash a \vee (a \supset \bot)} \quad a, \bot \vdash \bot}{a, (a \vee (a \supset \bot)) \supset \bot \vdash \bot} \qquad {\scriptstyle \supset_L}\dfrac{{\scriptstyle \vee_{RR}}\dfrac{{\scriptstyle \supset_R}\dfrac{{\scriptstyle \supset_L}\dfrac{a \vdash a \quad \bot, a \vdash \bot}{a \supset \bot, a \vdash \bot}}{a \supset \bot \vdash a \supset \bot}}{a \supset \bot \vdash a \vee (a \supset \bot)} \quad a \supset \bot, \bot \vdash \bot}{a \supset \bot, (a \vee (a \supset \bot)) \supset \bot \vdash \bot}}{a \vee (a \supset \bot), (a \vee (a \supset \bot)) \supset \bot \vdash \bot}}{a \vee (a \supset \bot) \vdash ((a \vee (a \supset \bot)) \supset \bot) \supset \bot}
$$

Strategy:
We conserve the existing proof theory properties . . .

Gentzen's major breakthrough (1930s):

- proofs can be analytic, *i.e.*, built in finitary ways,
- by horribly expensive algorithms,
- that nonetheless allow us to control and analyse them.

$$
\supset_{\mathsf{R}}\dfrac{\vee_{\mathsf{L}}\dfrac{\supset_{\mathsf{L}}\dfrac{\vee_{\mathsf{RL}}\dfrac{a \vdash a}{a \vdash a \vee (a \supset \bot)} \quad a, \bot \vdash \bot}{a, (a \vee (a \supset \bot)) \supset \bot \vdash \bot} \qquad \supset_{\mathsf{L}}\dfrac{\vee_{\mathsf{RR}}\dfrac{\supset_{\mathsf{R}}\dfrac{\supset_{\mathsf{L}}\dfrac{a \vdash a \quad \bot, a \vdash \bot}{a \supset \bot, a \vdash \bot}}{a \supset \bot \vdash a \supset \bot}}{a \supset \bot \vdash a \vee (a \supset \bot)} \quad a \supset \bot, \bot \vdash \bot}{a \supset \bot, (a \vee (a \supset \bot)) \supset \bot \vdash \bot}}{a \vee (a \supset \bot), (a \vee (a \supset \bot)) \supset \bot \vdash \bot}}{a \vee (a \supset \bot) \vdash ((a \vee (a \supset \bot)) \supset \bot) \supset \bot}
$$

But Gentzen

- only knew classical logic, which is poor for algorithms;
- only wanted finiteness, while we want more: efficiency;
- had no idea of proof complexity.

Strategy:

… while we keep proof complexity under control, …

Proof complexity = proof size (for propositional logic).

Strategy:

. . . while we keep proof complexity under control, . . .

Proof complexity $=$ proof size (for propositional logic).

Proof system $=$ algorithm that checks proofs in polynomial time.

Strategy:

... while we keep proof complexity under control, ...

Proof complexity = proof size (for propositional logic).

Proof system = algorithm that checks proofs in polynomial time.

Theorem [Cook & Reckhow(1974)]:

there exists a proof system whose proofs are all 'simple'

$$\leftrightarrow$$

$$\text{coNP} = \text{NP}$$

where 'simple' = verifiable in polynomial time over the size of the proved formula.

Strategy:

... while we keep proof complexity under control, ...

Proof complexity = proof size (for propositional logic).

Proof system = algorithm that checks proofs in polynomial time.

Theorem [Cook & Reckhow(1974)]:

there exists a proof system whose proofs are all 'simple'

$$\leftrightarrow$$

$$\text{coNP} = \text{NP}$$

where 'simple' = verifiable in polynomial time over the size of the proved formula.

So:

- we want to keep proof size low (and possibly making it lower),
- but not too low (otherwise we don't have proof systems).

Strategy:
. . . and we remove bureaucracy.

Idea: Let’s use the smallest conceivable bricks to build proofs.

(Inspired by Michelangelo, the idea is to remove the stone to find the statue, but we need a fine stone in the first place!)

Gentzen’s material is too rigid!

Strategy:
. . . and we remove bureaucracy.

Idea: Let's use the smallest conceivable bricks to build proofs.

(Inspired by Michelangelo, the idea is to remove the stone to find the statue, but we need a fine stone in the first place!)

Gentzen's material is too rigid!

We want proof systems whose inference steps are verifiable in constant time.

Example ('atomic cocontraction'):

$$\mathsf{m}\,\frac{\dfrac{a}{a \wedge a} \vee \dfrac{b}{b \wedge b}}{[a \vee b] \wedge [a \vee b]} \qquad \wedge \qquad \frac{a}{a \wedge a}$$

We call this property locality.

Problem: Are Two Given Proofs the Same?

- First formulated by Hilbert in 1900 [Thiele(2003)].
- Solutions depend on given criteria of 'sameness'.
- Solution:

$$\text{criterion} \quad \rightarrow \quad \text{decision procedure} \quad .$$

- Gentzen proof theory is not adequate precisely because its proofs are too coarse.
- So, the problem is embarrassingly open (but not for long, thanks to locality).

Problem: Are Two Given Proofs the Same?

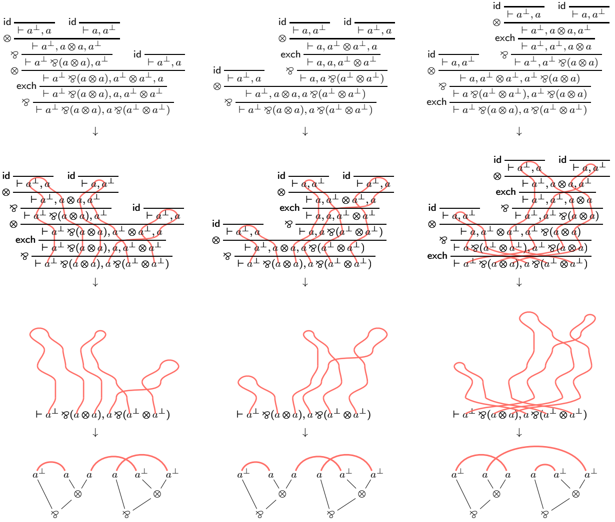
- First formulated by Hilbert in 1900 [Thiele(2003)].
- Solutions depend on given criteria of 'sameness'.
- Solution:

$$\text{criterion} \quad \to \quad \text{decision procedure} \quad .$$

- Gentzen proof theory is not adequate precisely because its proofs are too coarse.
- So, the problem is embarrassingly open (but not for long, thanks to locality).

BTW: Are two given algorithms the same?

Attempt in Gentzen Theory

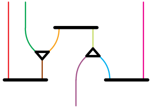


Picture taken from [Straßburger(2006)]

- From ‘different’ proofs we get proof nets [Girard(1987)],
- but they are too small (they probably are not a proof system).

Deep Inference and Atomic Flows (A Better Attempt)

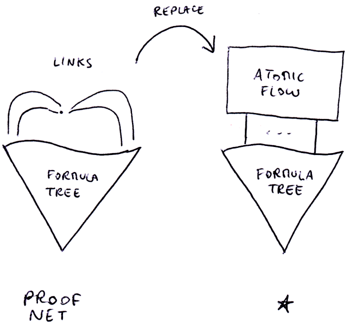
$$\mathsf{s}\,\frac{\mathsf{m}\,\dfrac{\dfrac{\mathsf{t}}{a\vee\bar a}}{[a\vee \mathsf{t}]\wedge[\mathsf{t}\vee\bar a]}}{\left[\mathsf{s}\,\dfrac{[a\vee \mathsf{t}]\wedge\bar a}{\dfrac{a\wedge\bar a}{\mathsf{f}}\vee \mathsf{t}}\vee \mathsf{t}\right]}
\qquad
=\frac{\left(\mathsf{s}\,\dfrac{a\wedge\left[\bar a\vee\dfrac{\mathsf{t}}{\bar a\vee a}\right]}{\dfrac{a\wedge\dfrac{\bar a\vee\bar a}{\bar a}}{\mathsf{f}}\vee\dfrac{a}{a\wedge a}}\wedge\bar a\right)}{a\wedge\dfrac{a\wedge\bar a}{\mathsf{f}}}
\qquad
\mathsf{m}\,\frac{\dfrac{a}{a\wedge a}\vee\dfrac{b}{b\wedge b}}{[a\vee b]\wedge[a\vee b]}\wedge\frac{a}{a\wedge a}$$


- Top row: deep inference proofs.
- Bottom row: (atomic) flows, extracted from the proofs above.
- Proofs composed by logical connectives: this yields locality.
- Atomic flows: logical info is lost and structural is kept.
- Flow size is polynomially related to derivation size.

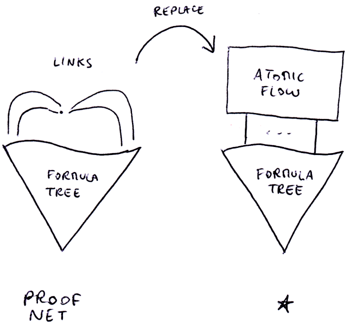
See [Guglielmi & Gundersen(2008)].

Conjecture



Conjecture: *(*) is a proof system.*

Conjecture



Conjecture: *(*) is a* *proof system.*

- This means that there should exist a polynomial algorithm to check the correctness of (*).
- If this is true, we have an excellent bureaucracy-free formalism.
- Note: if this were true of proof nets, then coNP $=$ NP.

Overview of Deep Inference Proof Systems

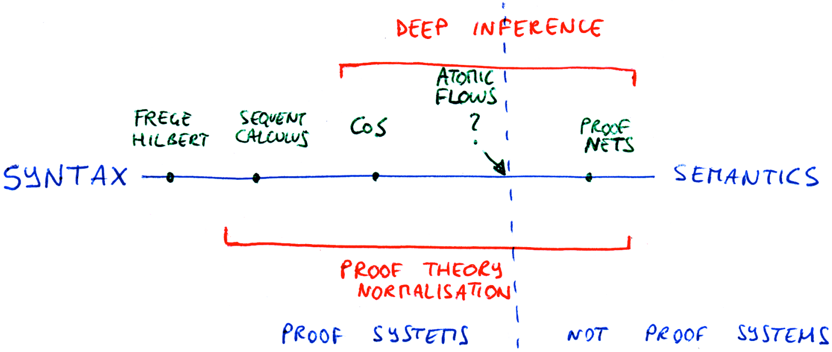
Started in 1999. All info in [Guglielmi(2010)].

There are now deep-inference proof systems for all logics:

- classical and intuitionistic;
- modal;
- linear;
- commutative, noncommutative and mixed.

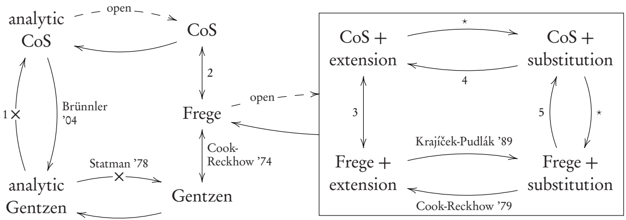
Locality can be achieved for all of these, and only in deep inference.

Elimination of Bureaucracy



Eliminate bureaucracy = find 'something' at the crossing.

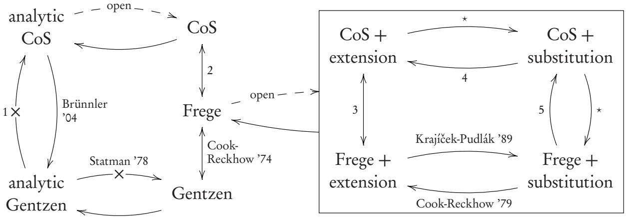
Are We Doing OK with Proof Complexity?



Short answer: yes.

$\longrightarrow$ means 'polynomially simulates'

Are We Doing OK with Proof Complexity?



Short answer: yes.

$\longrightarrow$ means 'polynomially simulates'

Deep inference has as small proofs as the best proof systems do

and

it has a normalisation theory

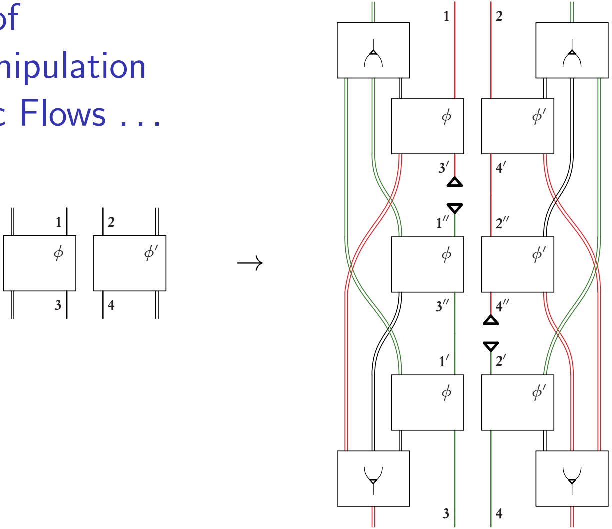
and

its analytic proof systems are more powerful than Gentzen ones

See

[Bruscoli et al.(2009)Bruscoli, Guglielmi, Gundersen, & Parigot].

Example of Proof Manipulation on Atomic Flows . . .



Even if there is a path between **1** and **3** on the left, there is none on the right (and the same for **2** and **4**).

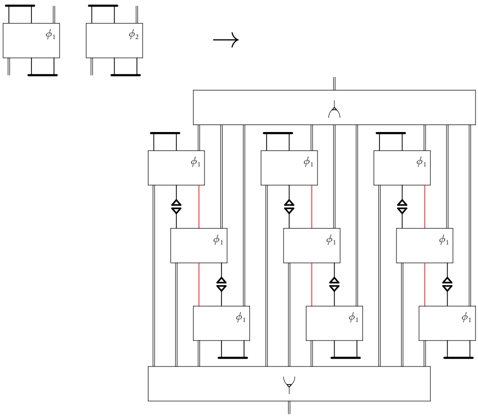
. . . and the Corresponding Proofs

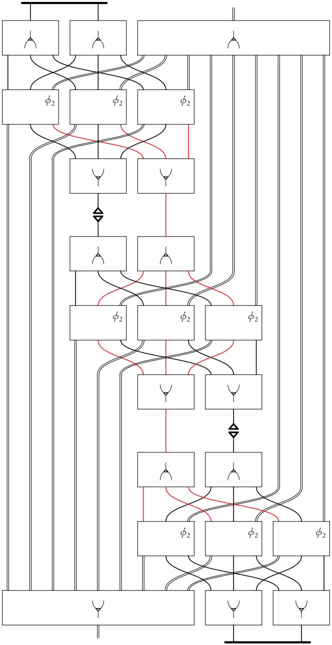
$$
\Phi = \begin{array}{c} [a \vee \bar{a}] \wedge \alpha \\ \Vert \\ \beta \vee (a \wedge \bar{a}) \end{array}
\quad \rightarrow \quad
\text{Break}\,\Phi =
\begin{array}{l}
\quad [a \vee \bar{a}] \wedge \dfrac{\alpha}{\alpha \wedge \alpha \wedge \alpha} \\
{}_{=}\overline{\qquad\qquad\qquad\qquad\qquad\qquad} \\
\left(\begin{array}{c} [a \vee \bar{a}] \wedge \alpha \\ {\scriptstyle\Phi}\Vert \\ \left[\beta \vee \left(\dfrac{a}{\mathsf{t}} \wedge \bar{a} \right) \right] \end{array} \wedge \alpha \wedge \alpha \right) \\
{}_{\mathsf{s}}\overline{\qquad\qquad\qquad\qquad\qquad\qquad} \\
\left(\left[\beta \vee \begin{array}{c} \left(\left[\dfrac{\mathsf{f}}{a} \vee \bar{a} \right] \wedge \alpha \right) \\ {\scriptstyle\Phi}\Vert \\ \left[\beta \vee \left(a \wedge \dfrac{\bar{a}}{\mathsf{t}} \right) \right] \end{array} \right] \wedge \alpha \right) \\
{}_{\mathsf{s}}\overline{\qquad\qquad\qquad\qquad\qquad\qquad} \\
\left[\beta \vee \beta \vee \begin{array}{c} \left(\left[a \vee \dfrac{\mathsf{f}}{\bar{a}} \right] \wedge \alpha \right) \\ {\scriptstyle\Phi}\Vert \\ \beta \vee (a \wedge \bar{a}) \end{array} \right] \\
{}_{=}\overline{\qquad\qquad\qquad\qquad\qquad\qquad} \\
\quad \dfrac{\beta \vee \beta \vee \beta}{\beta} \vee (a \wedge \bar{a})
\end{array}
$$

Only geometrical/topological structure matters.

Finding something like this is unthinkable without locality and atomic flows.

One More Example (Two Pieces)





Summary

Finding better ways of representing proofs

- The dream: proofs without unnecessary detail and even syntax
- The reality: lots of unnecessary detail and syntax
- Strategy: remove bureaucracy by keeping the good properties

The problem of proof identity

Exploiting locality

- Deep inference and atomic flows
- Eliminating bureaucracy in geometric proof systems
- Using geometry to manipulate proofs

Impact?

Impact

Wouldn't it be nice if all of maths ($\approx$ 100,000,000 pages) were represented as a semantic database?

We could:

- trust proofs (because they are automatically verified);
- access proofs at different abstraction levels (detail, just the idea, etc.);
- produce proofs by delegating routine tasks to the computer (with artificial intelligence?);
- ...

All fields of science will benefit.

Impact

Wouldn't it be nice if all of maths ($\approx$ 100,000,000 pages) were represented as a semantic database?

We could:

- trust proofs (because they are automatically verified);
- access proofs at different abstraction levels (detail, just the idea, etc.);
- produce proofs by delegating routine tasks to the computer (with artificial intelligence?);
- ...

All fields of science will benefit.

This will happen and it will be a REVOLUTION.

References

Bruscoli, P., Guglielmi, A., Gundersen, T., & Parigot, M. (2009).
Quasipolynomial normalisation in deep inference via atomic flows and threshold formulae.
`http://cs.bath.ac.uk/ag/p/QuasiPolNormDI.pdf`.

Cook, S., & Reckhow, R. (1974).
On the lengths of proofs in the propositional calculus (preliminary version).
In *Proceedings of the 6th annual ACM Symposium on Theory of Computing*, (pp. 135–148). ACM Press.

Girard, J.-Y. (1987).
Linear logic.
Theoretical Computer Science, *50*, 1–102.

Guglielmi, A. (2010).
Deep inference.
Web site at `http://alessio.guglielmi.name/res/cos`.

Guglielmi, A., & Gundersen, T. (2008).
Normalisation control in deep inference via atomic flows.
Logical Methods in Computer Science, *4*(1:9), 1–36.
`http://www.lmcs-online.org/ojs/viewarticle.php?id=341`.

Straßburger, L. (2006).
Proof nets and the identity of proofs.
Tech. Rep. 6013, INRIA.
`http://hal.inria.fr/docs/00/11/43/20/PDF/RR-6013.pdf`.

Thiele, R. (2003).
Hilbert's twenty-fourth problem.
American Mathematical Monthly, *110*, 1–24.