

A Dynamic Poincaré Principle

MANFRED KERBER

M.Kerber@cs.bham.ac.uk

<http://www.cs.bham.ac.uk/~mmk>

School of Computer Science
University of Birmingham, England




Overview



- 👉 Motivation – Proof vs Computation
- 👉 The Poincaré Principle
- 👉 A Dynamic Poincaré Principle
- 👉 Conclusion

Some Quotes


Some Quotes

 The only means to improve our conclusions is to make them as evident as those of the mathematicians are ... and when there is a dispute one needs only say: “Let’s calculate.” [Leibniz 1686]

Some Quotes


-  The only means to improve our conclusions is to make them as evident as those of the mathematicians are ... and when there is a dispute one needs only say: “Let’s calculate.” [Leibniz 1686]
-  Everything necessary for a correct inference is expressed in full ... nothing is left to guesswork. [Frege 1879]

Some More Quotes

 As a kind of dream I played (in 1968) with the idea of a future where every mathematician would have a machine on his desk, all for himself, on which he would write mathematics and which would verify his work. But, by lack of experience in such matters, I expected that such machines would be available in 5 years from then. But now, 23 years later, we are not that far yet.

[N.G. de Bruijn 1994]

Some More Quotes

 there is a very high degree of unexpectedness, combined with inevitability and economy. ... We do not want many 'variations' in the proof of a mathematical theorem: 'enumeration of cases', indeed, is one of the duller forms of mathematical argument. A mathematical proof should resemble a simple and clear-cut constellation, not a scattered cluster in the Milky Way. [Hardy 1940]

The de Bruijn Principle

A proof assistant satisfies the de Bruijn criterion if it has a proof checker that is small enough to be verified by hand. Proof assistants that have proof objects that are stored have the advantage of the possibility of independent checking.

A skeptical reader of a proof can take full control by checking the checker.

Overview

👉 Motivation – Proof vs Computation

👉 **The Poincaré Principle**

👉 A Dynamic Poincaré Principle

👉 Conclusion

The Poincaré Principle

Since computing is important for proving, one would like that if f is a computable function (on a freely generated algebra \mathcal{A}), then there is a formal expression $F(x)$ such that for all $a, b \in \mathcal{A}$

$$f(a) = b \quad \Leftrightarrow \quad \vdash F(\underline{a}) = \underline{b}, \quad (1)$$

for some representation $a \mapsto \underline{a}$ of elements of \mathcal{A} in the theory.

The most efficient way (from the point of proving) to ensure (1) is to add for each computable function f an expression $F(x)$ and postulate axiomatically that for arbitrary $a \in \mathcal{A}$

$$\vdash F(\underline{a}) = \underline{f(a)}. \quad (2)$$

[Barendregt 1997]

Example

Assume: $\forall n. n + 0 = n$ and $\forall n, m. n + s(m) = s(n + m)$

Prove: $P(3 + 1)$ and $\neg P(4)$ are contradictory.

1. $\forall n. n + 0 = n$ premise
2. $\forall n, m. n + s(m) = s(n + m)$ premise
3. $P(s(s(s(0))) + s(0))$ premise
4. $\neg P(s(s(s(s(0)))))$ premise
5. $\forall m. s(s(s(0))) + s(m) = s(s(s(s(0))) + m)$ $\forall e[n \mapsto s(s(s(0)))]$ 2
6. $s(s(s(0))) + s(0) = s(s(s(s(0))) + 0)$ $\forall e[m \mapsto 0]$ 5
7. $s(s(s(0))) + 0 = s(s(s(0)))$ $\forall e[n \mapsto s(s(s(0)))]$ 1
8. $s(s(s(0))) + s(0) = s(s(s(s(0))))$ =subst 7, 6
9. $P(s(s(s(s(0)))))$ =subst 8, 3
10. \perp $\neg e$ 9, 4

Much longer for $P(1,000,000 + 1,000,000)$.

Example (Cont'd)

- | | | |
|----|-------------|---------------|
| 1. | $P(3 + 1)$ | premise |
| 2. | $\neg P(4)$ | premise |
| 3. | $P(4)$ | calculation 1 |
| 4. | \perp | \neg e 3, 2 |

or even

- | | | |
|----|-------------|--------------------------------------|
| 1. | $P(3 + 1)$ | premise |
| 2. | $\neg P(4)$ | premise |
| 3. | \perp | \neg e _{calculation} 1, 2 |

Evaluation of the Poincaré Principle

- **Appropriate**, since it allows to take out trivial steps. Nobody wants to be bored by an uninformative proof.

Evaluation of the Poincaré Principle

- **Appropriate**, since it allows to take out trivial steps. Nobody wants to be bored by an uninformative proof.
- However, it is “**ad hoc**”, since it assumes a fixed notion of what is “trivial”.

Evaluation of the Poincaré Principle

- **Appropriate**, since it allows to take out trivial steps. Nobody wants to be bored by an uninformative proof.
- However, it is “**ad hoc**”, since it assumes a fixed notion of what is “trivial”.

Is the computation of 1,000,000 and 1,000,000 trivial?

Is the computation of $\frac{de^{\sin(\sqrt{x})}}{dx}$ trivial?

Evaluation of the Poincaré Principle

- **Appropriate**, since it allows to take out trivial steps. Nobody wants to be bored by an uninformative proof.
- However, it is “**ad hoc**”, since it assumes a fixed notion of what is “trivial”.
Is the computation of 1,000,000 and 1,000,000 trivial?
Is the computation of $\frac{de^{\sin(\sqrt{x})}}{dx}$ trivial?

Depends to a certain degree on human user.

Overview

👉 Motivation – Proof vs Computation

👉 The Poincaré Principle

👉 **A Dynamic Poincaré Principle**

👉 Conclusion

Dynamic Poincaré Principle

At any point in a proof it is possible to introduce an argument “calculation”, which a reader can deal with in three different ways:

- (1) Believe that the computation is correct.*
- (2) Check a proof that the algorithm performing the computation is correct.*
- (3) Check that a trace of the concrete computation is correct.*

Example

Assume a program **DIV**:

```
a := 0 ;  
b := x ;  
while b >= y do  
    b := b - y;  
    a := a + 1  
end
```

We can prove in the Hoare calculus:

$$\{x \geq 0 \wedge y > 0\} \text{ DIV } \{a \cdot y + b = x \wedge 0 \leq b < y\}.$$

Example (Cont'd)

If we want to prove

Hyp	$P(6/3)$	Hyp
Thm	$P(2)$???

then there is mainly one standard argument. Namely we argue that we use a calculation to compute $6/3$.

Hyp	$P(6/3)$	Hyp
Thm	$P(2)$	calculation

Example (Cont'd)

'calculation' is not a primitive explanation, hence there are three possible reactions by the user:

(1) **Believe** that the computation is correct.

Example (Cont'd)

'calculation' is not a primitive explanation, hence there are three possible reactions by the user:

- (1) **Believe** that the computation is correct.
- (2) **Check** a proof that the **algorithm** performing the computation is correct.
(Inspect the proof in the Hoare calculus)

Example (Cont'd)

'calculation' is not a primitive explanation, hence there are three possible reactions by the user:

- (1) **Believe** that the computation is correct.
- (2) **Check** a proof that the **algorithm** performing the computation is correct.

(Inspect the proof in the Hoare calculus)

- (3) **Check** that **a trace** of the concrete computation is correct.

Hyp	$P(6/3)$	Hyp
1	$6/3 = s((6 - 3)/3)$	DefDiv
3	$6/3 = s(s(((6 - 3) - 3)/3))$	DefDiv
2	$6/3 = s(s((3 - 3)/3))$	DefMinus (With tactics)
4	$6/3 = s(s(0/3))$	DefMinus
5	$6/3 = s(s(0))$	DefDiv
Thm	$P(2)$	Subst=

DefDiv and DefMinus which may be further expanded, see e.g., [KerberKohlhaseSorge 1998])

Why different expansions?

Different proofs have different advantages. A high-level proof will capture the essential parts, it is more concise and may be easily generalized. On this level, the proof

Hyp $P(6/3)$ Hyp
Thm $P(2)$ calculation

is the same as the proof

Hyp $P(1,000,000/2)$ Hyp
Thm $P(500,000)$ calculation

while the explicit descriptions on a low logical level are quite different (for instance, they have significantly different lengths).

Overview

👉 Motivation – Proof vs Computation

👉 The Poincaré Principle

👉 A Dynamic Poincaré Principle

👉 **Conclusion**

Conclusion

- The Poincaré Principle is important since it allows for more abstract proofs.
- It is to a certain degree “ad hoc”, since it assumes a fixed notion of “trivial”. It can be extended to a dynamic principle which puts it in the hands of a human user whether to expand an argument in different ways.