the morning paper

an interesting/influential/important paper from the world of CS every weekday morning, as selected by Adrian Colyer

How to write a 21st Century Proof

JANUARY 12, 2015

How to write a 21st Century Proof (http://research.microsoft.com/en-us/um/people/lamport/pubs/proof.pdf) – Lamport 2012

In this paper Leslie Lamport shares what he has learned in well over 20 years of writing proofs.

> *I am a computer scientist who was educated as a mathematician. I discovered structured proofs through my work on concurrent (multiprocess) algorithms. These algorithms can be quite subtle and hard to get right; their correctness proofs require a degree of precision and rigor unknown to most mathematicians (and many computer scientists). A missing hypothesis, such as that a set must be nonempty, which is a trivial omission in a mathematical theorem, can mean a serious bug in an algorithm.*

The paper is aimed at mathematicians, showing how the approach to proofs that Lamport developed can also be applied in a pure mathematical context. Mathematicians normally use only informal prose to demonstrate their proofs. A working example through the paper is based on a proof from a celebrated mathematics text book ***Calculus*** **(http://www.cambridge.org/gb/academic/subjects/mathematics/real-and-complex-analysis/calculus-3rd-edition?format=HB)** by Spivak.

> *When I started writing structured proofs, I quickly found them to be completely natural. Writing non-trivial prose proofs now seems as archaic to me as writing "The number e raised to the power of i times π, when added to 1, equals 0."*

Mathematicians, Lamport claims, have been writing proofs pretty much the same way since the 17th century, and

> *the important goal is to stop writing 17th century prose proofs in the 21st century.*

Lamport makes proofs easier to understand by the introduction of rigour in structure and in naming: structure to formally identify the roles of statements and how they connect together; naming to make it clear which facts are being used and refered to. Such structuring Lamport says makes it *possible* to avoid errors, but hard work is still needed to make it *probable*! How can we increase our chances?

> *The best way I know to eliminate errors is to imagine that there is a curious child sitting next to us. Every time we write an assertion, the child asks: Why?*

We also need to be aware of our own psychological bias:

> *When you write a proof, you believe the theorem to be true. The only way to avoid errors is to be ruthlessly suspicious of everything you believe. Otherwise, your natural desire to confirm what you already believe to be true will cause you to miss gaps in the proof; and every gap could hide an error that makes the entire result wrong.*

The paper contains an incremental re-working of a proof from Spivak's text book into a more rigourous format. Along the way, several unstated implications and pre-requisities in the original come to light and are addressed.

We end up with a short look at TLA+, the language that Lamport designed for structured proofs in an CS context. Previously in this series we saw how **TLA+ has been used to great effect at Amazon (https://blog.acolyer.org/2014/11/24/use-of-formal-methods-at-amazon-web-services/)**.

Structure in TLA+ comes from a number of statements (this paper highlights a subset useful to mathematicians):

> ASSUME/PROVE and NEW – where NEW introduces a new variable with conditions, ASSUME states the set of facts 'in scope' for a subsequent proof, and PROVE shows what is to be proved (the goal of that step).
> SUFFICES A, which is an assertation that proving A proves the current goal (see also the **TLA+ guide (http://research.microsoft.com/en-us/um/people/lamport/tla/tla2-guide.pdf)**, p25).
> PICK as in 'pick some x in the set S' introduces a new variable x and asserts that some statement regarding that x must be true – the steps proof must show this to be the case.
> CASE – for structuring case-by-case reasoning

A careful hierarchical numbering scheme is introduced which is designed to prevent 'illegal' references to sub-proofs which may have been proved under assumptions that differ from those currently in scope. If you think about variable scoping rules you'll quickly get the idea.

Comments can be distracting in the middle of a proof ('in arbitrary places') according to Lamport, but a proof sketch between a statement and its proof works very well.

> *Proof sketches can be used at any level in a hierarchical proof, including before the highest-level proof. A reader not interested in the details of that part of the proof*
> *can read just the proof sketch and skip the steps and their proofs.*

See the **TLA+ guide (http://research.microsoft.com/en-us/um/people/lamport/tla/tla2-guide.pdf)** for a full overview of TLA+.

> *Mathematicians think that the logic of the proofs they write is completely obvious, but our examination of Spivak's proof shows that they are wrong. Students are expected to learn how to write logically correct proofs from examples that, when read literally, are illogical. (Recall the first sentence of Spivak's proof.) It is little wonder that so few of them succeed. Learning to write structured formal proofs that a computer can check will teach students what a proof is. Going from these formal proofs to structured informal proofs would then be a natural step.*

So, how should one begin?

> *Fortunately, it's not hard to write 21st century proofs. There is no need to wait until other mathematicians are doing it. You can begin by just adding structure to an existing proof… Start by rewriting a simple proof and then try longer ones. You should soon find this a much more logical way to write your proofs, and readers will have no trouble understanding the proof style.*

And since many readers of this blog are probably concerned with computer science first, and mathematics second if at all, here's what Lamport has to say about the benefits of the technique in the context of algorithms:

> *Proofs of algorithms are most often mathematically shallow but complicated, requiring many details to be checked. With traditional prose proofs, I found it impossible to make sure that I had not simply forgotten to check some detail. Computer science offers a standard way to handle complexity: hierarchical structure. Structured proofs were therefore an obvious solution. They worked so well for proofs of algorithms that I tried them on the more mathematical proofs that I write. I have used them for almost every proof of more than about ten lines that I have published since 1991.*

It would be an interesting exercise to attempt a TLA+ proof of the informal prose proof of schema evolution safety given in the **Google F1 schema paper (https://blog.acolyer.org/2015/01/07/online-aysnchronous-schema-change-in-f1/)**.

from → Uncategorized

5 Comments leave one →

1. **Dave PERMALINK**
   January 12, 2015 7:23 am
   Isn't this very similar to the work that Vladimir Voevodsky is doing. See the "Alternative Logic" section of this wired article: http://www.wired.com/2013/03/computers-and-math/all/

   REPLY
   - **adriancolyer PERMALINK***
     January 12, 2015 11:41 am
     That's a very interesting piece, thanks for posting. There is definitely a similarity, although Lamport's emphasis in this paper is not so much on automated proofs, as on a methodology by which mathematicians could make their on-paper arguments clearer and more robust. The TLA+ Proof System (TLAPS) https://tla.msr-inria.inria.fr/tlaps/content/Home.html can be used to automate checking of some TLA+ proofs though.

     REPLY

Trackbacks

1. The Part-Time Parliament | the morning paper
2. In Search of an Understandable Consensus Algorithm | the morning paper
3. An empirical study on the correctness of formally verified distributed systems | the morning paper