Основы формальной верификации

с помощью интерактивных доказателей теорем

Тимофей Черганов
cherganov@gmail.com

01.06.2025

Введение

Корректна ли программа?

Введение

Корректна ли программа?

- Therac-25 – ошибка в программном обеспечении аппарата лучевой терапии, которая привела к гибели по меньшей мере 5 человек.
- Ariane 5 Disaster, Mars Climate Orbiter, Mariner 1, Patriot missile
- The Pentium bug
- The DAO Attack

Введение

Обычно используют:

- ревью кода,
- различные виды тестирования,
- статические анализаторы.

Введение

Обычно используют:

- ревью кода,
- различные виды тестирования,
- статические анализаторы.

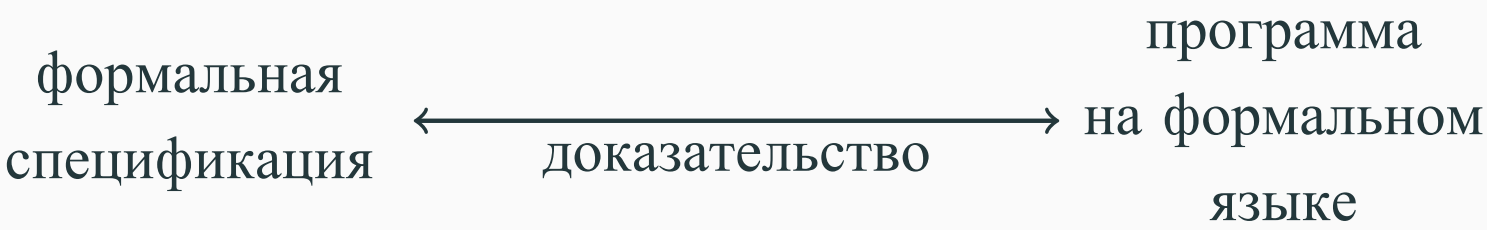
Не могут гарантировать корректность программ в общем случае.

Введение

Корректна ли программа?

Введение

Корректна ли программа?



Доказательство

Доказательство – это дерево вывода в некоторой формальной системе.

$$\dfrac{\dfrac{\phi \qquad \psi}{(\phi \wedge \psi)}\,(\wedge\mathrm{I}) \qquad \chi}{((\phi \wedge \psi) \wedge \chi)}\,(\wedge\mathrm{I})$$

Формальная система:

- синтаксис (разрешенные символы и как они образуют выражения),
- аксиомы,
- правила вывода.

Доказательство

Примеры формальных систем: FOL + ZFC, TG, HOL, Martin-Löf type theory.

Доказательство

Примеры формальных систем: FOL + ZFC, TG, HOL, Martin-Löf type theory.

Формальная система должна позволять автоматически проверять полученные доказательства (например, Agda, Coq, Lean, HOL4, HOL Light, Isabelle/HOL, Mizar).

Теория типов

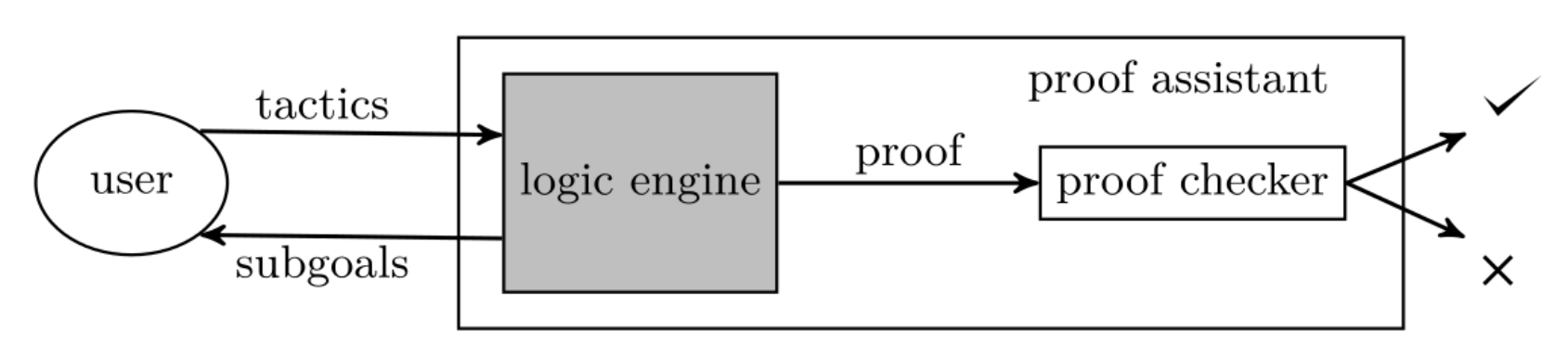
История:

1908	Bertand Russell. Mathematical logic based on the theory of types. $X := \{A \mid A \notin A\}$. $X \in X$?
1930s	simple type theory (Carnap, Ramsey, Quine, Tarski)
1940	Alonzo Church. A formulation of the simple theory of types
1969	William Howard. The formulae-as-types notion of construction
1979	Per Martin-Löf. Constructive mathematics and computer programming
2010	Vladimir Voevodsky. Univalent foundations project

Подробнее:

- `https://plato.stanford.edu/entries/type-theory`
- `https://ncatlab.org/nlab/show/type+theory`
- Nederpelt, Geuvers. *Type Theory and Formal Proof: An Introduction.*
- *Homotopy Type Theory: Univalent Foundations of Mathematics.* `https://homotopytypetheory.org/book`

Интерактивные доказатели теорем



Интерактивные доказатели теорем

История:

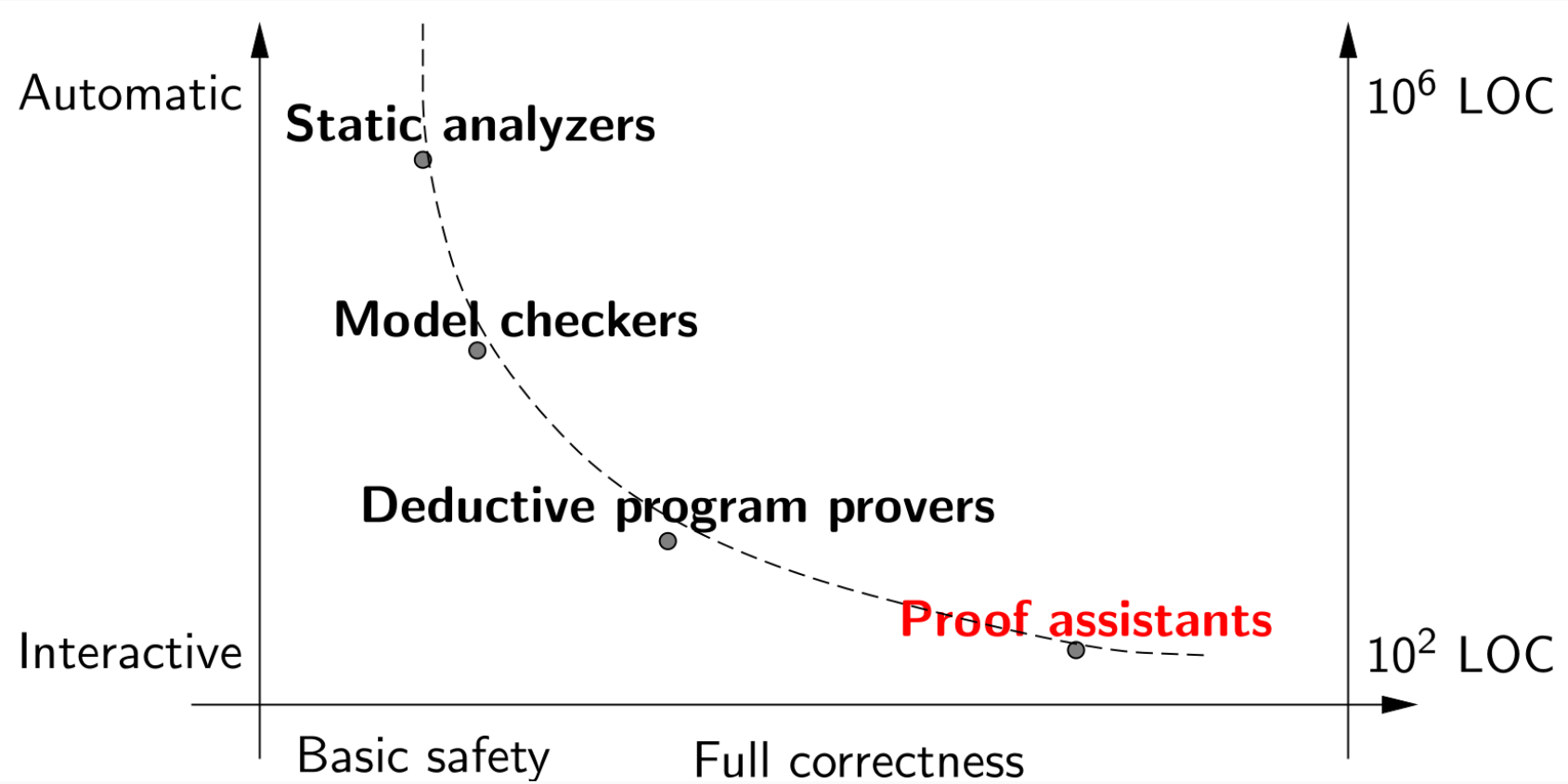
1968 Nicolaas de Bruijn. Automath (Algol 60)
1972 Robin Milner. Stanford LCF (ML)
1978 Michael Gordon. Edinburgh LCF (ML)
1984 Robert Constable. NuPRL (Common Lisp)
1986 Isabelle (Standard ML)
1988 Mike Gordon. HOL88 (ML)
1989 Thierry Coquand. Coq (OCaml)
1999 Catarina Coquand. Agda (Haskell)
2013 Leonardo de Moura. Lean (Lean, C++)

Подробнее:

Harrison, Urban, Wiedijk. *History of Interactive Theorem Proving.*

`https://www.cl.cam.ac.uk/~jrh13/papers/joerg.pdf`

Интерактивные доказатели теорем



Интерактивные доказатели теорем

Although the proof may have the size of several Megabytes, the verifying program can be small. This program then can be inspected in the usual way by a mathematician or logician. If someone does not believe the statement that a proof has been verified, one can do independent checking by a trusted proof-checking program. In order to do this one does need formal proofs of the statements. A Mathematical Assistant satisfying the possibility of independent checking by a small program is said to satisfy the *de Bruijn criterion*.

Barendregt, Wiedijk. *The Challenge of Computer Mathematics.*

`https://www.cs.ru.nl/~freek/notes/RSpaper.pdf`

Примеры использования интерактивных доказателей

- формализация математики (Four-color theorem, Feit-Thompson theorem, Homotopy type theory, …),
- теория языков программирования,
- корректность программ (CompCert, seL4, FSCQ, Fiat-Crypto, …)

Примеры использования интерактивных доказателей

CompCert – компилятор C, написанный и формально верифицированный на Coq

The striking thing about our CompCert results is that the middle-end bugs we found in all other compilers are absent. As of early 2011, the under-development version of CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task. The apparent unbreakability of CompCert supports a strong argument that developing compiler optimizations within a proof framework, where safety checks are explicit and machine-checked, has tangible benefits for compiler users.

Yang, Chen, Eide, Regehr. *Finding and Understanding Bugs in C Compilers.*

`https://users.cs.utah.edu/~regehr/papers/pldi11-preprint.pdf`

Примеры использования интерактивных доказателей

CakeML is a functional programming language and an ecosystem of proofs and tools built around the language. The ecosystem includes a proven-correct compiler that can bootstrap itself.

The source and proofs for CakeML are developed in the HOL4 theorem prover.

Примеры использования интерактивных доказателей

seL4 – микроядро операционной системы.

Formal proof of functional correctness, security properties (integrity, confidentiality), and absence of specific runtime errors (done in Isabelle/HOL).

Примеры использования интерактивных доказателей

В проекте CertiKOS показан композиционный подход к построению формально верифицированных многопоточных ядер ОС. Многопоточность допускает одновременное выполнение пользовательского и ядерного кода на разных уровнях абстракции. Каждый такой уровень может иметь свой набор наблюдаемых событий. В работе проводится формальная спецификация этих уровней и их наблюдаемых событий, а затем каждый модуль ядра формально верифицируется на своем уровне абстракции. Для разработки и верификации функциональной корректности многопоточного ядра используется Coq.

Дополнительные преимущества

1. Заставляет разработчиков системно подходить к проблемам
2. Улучшает качество спецификаций даже без формальной верификации
3. Приводит к лучшему дизайну
4. Обеспечивает строгую документацию внутри команды разработчиков
5. Предоставляет основу для повторного использования через сопоставление спецификаций (specification matching)
6. Может заменить необходимость выполнения большого количества тестов

План

- основы Coq;
- верификация функциональных программ (на Coq);
- верификация императивных программ (на C).

Coq

Coq[1] – это система для работы с формальными доказательствами. Она предоставляет язык для записи математических определений, алгоритмов и теорем, а также среду для интерактивного построения доказательств.

Coq можно установить[2] на Windows, macOS и Linux. Для Visual Studio Code, Emacs и Vim существуют расширения[3], которые поддерживают работу с Coq.

---

[1] `https://coq.inria.fr`
[2] `https://coq.inria.fr/download`
[3] `https://coq.inria.fr/user-interfaces.html`

Установка

Самый простой способ установки – это использовать пакетный менеджер opam `https://opam.ocaml.org`.

Пример установки на Ubuntu:

```
apt install opam
opam init
opam pin add coq 8.20.1
```

Для работы в vscode нужно установить language server и расширение VsCoq (по умолчанию должна установиться версия 2.2.6):

```
opam install vscoq-language-serve
```

Литература

Основная литература:

1. Benjamin C. Pierce et al. *Software Foundations*. `https://softwarefoundations.cis.upenn.edu`
2. The Coq Development Team. *The Coq Reference Manual*. `https://coq.inria.fr/refman`

Дополнительная литература:

1. Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Springer, 2004. `https://www.labri.fr/perso/casteran/CoqArt/index.html`
2. Adam Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. MIT Press, 2013. `http://adam.chlipala.net/cpdt`
3. Talia Ringer, Karl Palmskog, Ilya Sergey, Milos Gligoric, and Zachary Tatlock. *QED at Large: A Survey of Engineering of Formally Verified Software*. Foundations and Trends in Programming Languages: Vol. 5: No. 2-3, pp 102-281, 2019. `https://arxiv.org/abs/2003.06458`

Сообщество

`https://coq.inria.fr/community`

На русском языке:

- Формальные методы верификации ПО на практике
  `https://t.me/practical_fm`
- Зависимые типы в массы!
  `https://t.me/joinchat/Ai4h2D9SWO8GfISyv-CHsQ`
- Coq
  `https://t.me/+NOQg8UpVDRE1MWNi`