Systemy dowodzenia twierdzeń

Iakymenko Kyrylo, Mróz Michał 13 czerwca 2024

Spis treści

Czym są systemy dowodzenia twierdzeń?

Systemy służące do dowodzenia twierdzeń

Rodzaje systemów dowodzenia twierdzeń

Istnieją dwa główne rodzaje systemów tego typu:

- CIC - Calculus of Inductive Constructions.
- LCF-Style

Schemat działania systemu w stylu LCF

W systemie zdefiniowany jest typ dowodu, oraz zbiór funkcji przekształcających ten typ. Funkcje te odpowiadają regułom inferencyjnym.

"Formalizacja 100 twierdzeń"

99%

The page does not keep track of *all* formalizations of these theorems. It just shows formalizations in systems that have formalized a significant number of theorems, or that have formalized a theorem that none of the others have done. The systems that this page refers to are (in order of the number of theorems that have been formalized, so the more interesting systems for mathematics are near the top):

Isabelle	90
HOL Light	87
Coq	79
Lean	76
Metamath	74
Mizar	69
nqthm/ACL2	45
ProofPower	43
PVS	26
Megalodon	12
Naproche	10
NuPRL/MetaPRL	8

"Formalizacja 100 twierdzeń"

21. **Green's Theorem**
    Isabelle, Mohammad Abdulaziz & Larry Paulson: statement

22. **The Non-Denumerability of the Continuum**
    Isabelle, Benjamin Porter: statement
    HOL Light, John Harrison: statement
    Coq, C-CoRN: statement
    Lean, Floris van Doorn: statement
    Metamath, Norman Megill: statement
    Mizar, Grzegorz Bancerek: statement
    ACL2, Ruben Gamboa & John Cowles
    ProofPower, Rob Arthan: statement
    PVS, Clément Blaudeau
    Megalodon: statement

"Formalizacja 100 twierdzeń"

5. **Prime Number Theorem**
   Isabelle, Jeremy Avigad et al.: statement
   HOL Light, John Harrison: statement
   Metamath, Mario Carneiro: statement

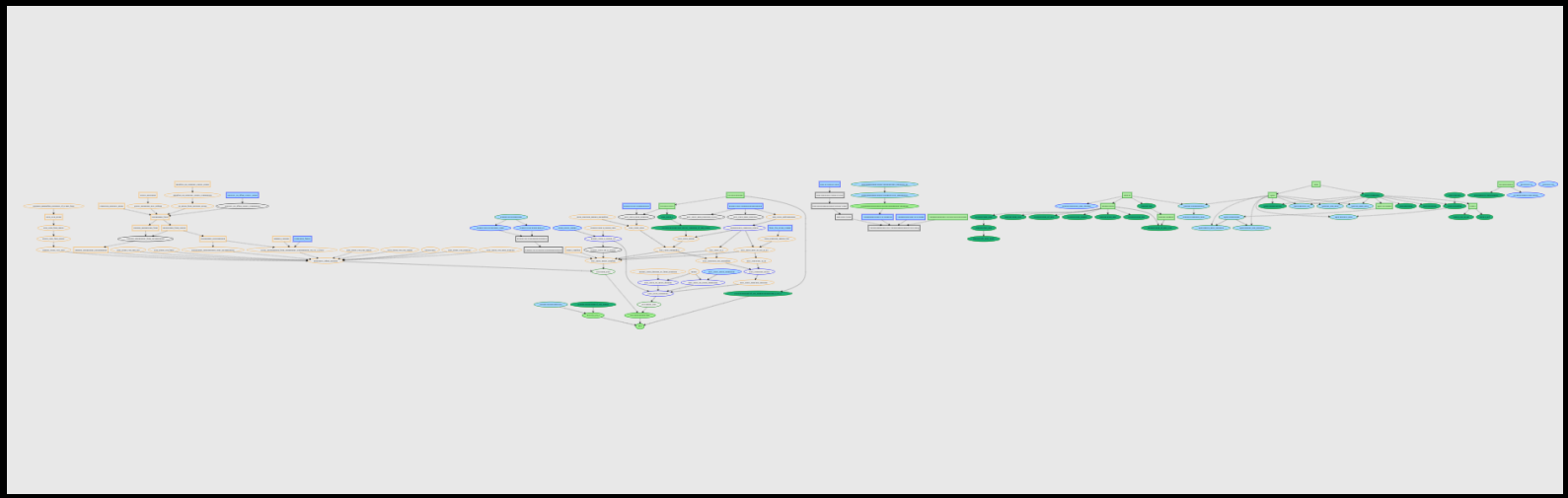
6. **Gödel's Incompleteness Theorem**
   Isabelle, Larry Paulson: statement
   HOL Light, John Harrison: statement
   Coq, contrib, Russell O'Connor: statement
   nqthm, Natarajan Shankar

"Formalizacja 100 twierdzeń"

33. *Fermat's Last Theorem*

"Formalizacja 100 twierdzeń"

"Formalizacja 100 twierdzeń"



Jakie istnieją systemy?

Porównanie systemów			
Nazwa	Rok i twórca	Typ	100 twierdzeń
Coq	1989 INRIA	CiC	79
Lean	2013 Leonardo de Moura	CiC	76
HOL (Light)	1988 Michael Gordon	LCF	87
Isabelle	1986 Lawrence Paulson	LCF	90
Agda	2007 Ulf Norell	CiC	-
Metamath	2005 Norman Megill	-	74
Mizar	1973 Andrzej Trybulec	-	69

Zastosowania

Systemy tego typu wykorzystywane są między innymi w dowodzeniu poprawności działania koprocesorów[1]. Innym przykładem może być dowód twierdzenia o czterech barwach[2].

Zastosowania

seL4[3] - mikrojądro systemowe, którego podstawowe funkcje były formalnie zweryfikowane. Oznacza to, że jego podstawowe funkcje (komunikacja międzyprocesowa, zarządzanie pamięcią itp.) są zgodne ze specyfikacją. Dowód formalny został przeprowadzony w Isabelle/HOL.

Rola typów w dowodzeniu twierdzeń

Zdania logiczne = Typy

Propositions as types[4]

Rola typów w dowodzeniu twierdzeń

```
let conj_def =
  prim_new_const_definition ("/\\",
  parse_term("\\p1 p2. !p. (p1 ==> (p2 ==> p)) ==> p"));;
```

Rola typów w dowodzeniu twierdzeń

```
,,
-( 02:16:31 )-< command 51 >-------------------------------------------{ counter: 0 }-
utop # conj_def;;
- : thm =
Theorem ([],
 Tmcomb
  (Tmcomb
    (Tmconst ("=",
      Tycomp ("->",
       [Tycomp ("->",
         [Tycomp ("bool", []);
          Tycomp ("->", [Tycomp ("bool", []); Tycomp ("bool", [])])]);
        Tycomp ("->",
         [Tycomp ("->",
           [Tycomp ("bool", []);
            Tycomp ("->", [Tycomp ("bool", []); Tycomp ("bool", [])])]);
          Tycomp ("bool", [])])])),
    Tmconst ("/\\",
     Tycomp ("->",
      [Tycomp ("bool", []);
       Tycomp ("->", [Tycomp ("bool", []); Tycomp ("bool", [])])]))),
  Tmabs (Tmvar ("p1", Tycomp ("bool", [])),
   Tmabs (Tmvar ("p2", Tycomp ("bool", [])),
    Tmcomb
     (Tmconst ("!",
```

Rola typów w dowodzeniu twierdzeń

Dowód twierdzenia polega na skonstruowaniu wyrażenia, które ma odpowiedni typ.

HOL Zero

HOL Zero[5] - prosty system dowodzenia twierdzeń stworzony jako narzędzie pedagogiczne. Z uwagi na ogólnodostępny kod (pod licencją GNU GPL 3+) i kompleksową dokumentację, został wykorzystany przez nas jako wzór systemu.

Początkowe trudności

Zauważyliśmy, że Scala ma bardziej restrykcyjny system typowania niż język OCaml. Powodowało to początkowo problemy ze zgodnością typów, szczególnie w sytuacji w której łączyliśmy funkcje zdefiniowane na typach generycznych z funkcjami operującymi na konkretnych typach.

Podejście funkcyjne i obiektowe

Podejście obiektowe umożliwia nam pisanie kodu z wewnętrzną strukturą (klasy, dziedziczenie, interfejsy itp.)
Podejście funkcyjne pozwala na wygodną prace z kodem na wyższym poziomie abstrakcji.

Testowanie programu

Wykorzystanie scalatest dało nam możliwość pisania przejrzystych testów, co było dużą pomocą, w sytuacji w której oryginalny kod był testów pozbawiony.
Narzędziem to umożliwia pisanie testów językiem naturalnym, co sprawia, że komunikaty o niepowodzeniu są zazwyczaj łatwe do zrozumienia.

Rozbudowane biblioteki

Oryginalny projekt napisany był w języku OCaml, co sprawiało, że konieczne było zdefiniowanie większości prostych funkcji. Przykładowo funkcje zdefiniowane dla list:

```
    let is_empty xs =
        match xs with
            [] -> true
        | _  -> false;;
```

Można zastąpić wywołaniem pojedynczej metody listy.

Dziękujemy za uwagę

[1] John Harrison. *Automated Theorem Proving in Real Applications*. Tech. rep. Intel Corporation, 2002.

[2] Georges Gonthier. “Formal Proof - The four-Color Theorem”. In: *Notices of the AMS* (2008).

[3] “seL4”.

[4] Philip Wadler. “Propositions as Types”.

[5] proof-technologies. *HOL Zero*. URL: http://www.proof-technologies.com/holzero/index.html.