

Comparison of two theorem provers: Isabelle & Coq

A. Yushkovskiy S. Tripakis

Department of Computer Science
School of Science
Aalto University

CS-E4000: Seminar in Computer Science
autumn 2017

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Introduction

History of logic

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

PRINCIPLES OF MATHEMATICAL LOGIC

BY

D. HILBERT AND W. ACKERMANN

TRANSLATED FROM THE GERMAN BY

LEWIS M. HAMMOND • GEORGE G. LECKIE • F. STEINHARDT

Professor of Philosophy
University of Virginia

Professor of Philosophy
Emory University

Columbia University

EDITED AND WITH NOTES BY
ROBERT E. LUCE
Assistant Professor of Mathematics
Rutgers University

PUBLISHED AND DISTRIBUTED IN THE PUBLIC INTEREST BY AUTHORITY
OF THE ATTORNEY GENERAL UNDER LICENSE NUMBER A-1428

Über die Bedeutung des Satzes vom ausgeschlossenen Dritten in der Mathematik, insbesondere in der Funktionentheorie ¹⁾.

Von L. E. J. Brouwer in Amsterdam.

§ 1.

Innerhalb eines bestimmten endlichen „Hauptsystems“ können Eigenschaften von Systemen, d. h. Abbildbarkeiten von Systemen auf andere Systeme mit vorgeschriebenen Elementkorrespondenzen, immer *geprüft* (d. h. entweder bewiesen oder ad absurdum geführt) werden; die durch die betreffende Eigenschaft angewiesene Abbildung besitzt nämlich auf jeden Fall nur eine endliche Anzahl von Ausführungsmöglichkeiten, von denen jede für sich unternommen und entweder bis zur Beendigung oder bis zur Hemmung fortgesetzt werden kann. (Hierbei liefert das Prinzip der mathematischen Induktion oft das Mittel, derartige Prüfungen ohne individuelle Betrachtung jedes an der Abbildung beteiligten Elementes bzw. jeder für die Abbildung bestehenden Ausführungsmöglichkeit durchzuführen; demzufolge kann die Prüfung auch für Systeme mit sehr großer Elementenzahl mitunter verhältnismäßig schnell verlaufen.)

Auf Grund der obigen Prüfbarkeit gilt für innerhalb eines bestimmten endlichen Hauptsystems konzipierte Eigenschaften der *Satz vom ausgeschlossenen Dritten*, d. h. das Prinzip, daß jede Eigenschaft für jedes System entweder richtig oder unmöglich ist, und insbesondere der *Satz von der Reziprozität der Komplementärspezies*, d. h. das Prinzip, daß für jedes System aus der Unmöglichkeit der Unmöglichkeit einer Eigenschaft die Richtigkeit dieser Eigenschaft folgt.

Wenn z. B. die Vereinigung $\mathfrak{S}(p, q)$ zweier mathematischer Spezies p und q wenigstens 11 Elemente enthält, so folgt hieraus auf Grund des (in diesem Falle

Introduction

History of logic

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary



EDITED AND WITH NOTES BY
ROBERT E. LUCE
Assistant Professor of Mathematics
Rutgers University

PUBLISHED AND DISTRIBUTED IN THE PUBLIC INTEREST BY AUTHORITY
OF THE ATTORNEY GENERAL UNDER LICENSE NUMBER A-1428

Übe

von S
gesch
oder
wiese
Ausfu
bis z
liefert

tionen ohne individuelle Betrachtung jedes an der Abbildung beteiligten Elementes bzw. jeder für die Abbildung bestehenden Ausführungsmöglichkeit durchzuführen; demzufolge kann die Prüfung auch für Systeme mit sehr großer Elementenzahl mitunter verhältnismäßig schnell verlaufen.)

Auf Grund der obigen Prüfbarkeit gilt für innerhalb eines bestimmten endlichen Hauptsystems konzipierte Eigenschaften der *Satz vom ausgeschlossenen Dritten*, d. h. das Prinzip, daß jede Eigenschaft für jedes System entweder richtig oder unmöglich ist, und insbesondere der *Satz von der Reziprozität der Komplementärspezies*, d. h. das Prinzip, daß für jedes System aus der Unmöglichkeit der Unmöglichkeit einer Eigenschaft die Richtigkeit dieser Eigenschaft folgt.

Wenn z. B. die Vereinigung $\mathfrak{S}(p, q)$ zweier mathematischer Spezies p und q wenigstens 11 Elemente enthält, so folgt hieraus auf Grund des (in diesem Falle



Outline

Foundations of Formal Approach

- A formal system

- Classical and Intuitionistic logics

Isabelle & Coq

- First acquaintance

- Basic syntax

- Example of proof

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Elements of a Formal System

- A formula (judgement, statement) $\phi \in \Phi$:

$$\begin{aligned}\phi &:= p \mid q \mid \dots \\ &\mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \neg \phi_1 \mid \phi_1 \rightarrow \phi_2 \\ &\mid \text{true} \mid \text{false} \\ &\mid \dots\end{aligned}$$

- Propositional variables: — $p, q, \dots \in V$
- An axiom — $\phi_A \in A$
- An inference rule τ — a transition function $\tau : \Phi \rightarrow \Phi$
- A formula ϕ provable from Φ — $\Phi \vdash \phi$
- A tautology \top — $\vdash \phi$
- A contradiction \perp — $\vdash \neg \phi$

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Definition of the formal system

A *formal system* is a quadruple $\Gamma = \langle A, V, \Omega, R \rangle$, where

- A – set of axioms
- V – set of propositional variables
- Ω – set of logical operators
- R – set of inference rules

A *formal proof* of the formula ϕ is a finite sequence of judgements $\psi_1 \xrightarrow{\tau_1} \psi_2 \xrightarrow{\tau_2} \dots \xrightarrow{\tau_n} \psi_n$, where $\psi_i \in A \cup \{\psi_k\}_{k=1}^{i-1}$

Types of written proofs:

- backward proof: $goals \xrightarrow{\tau} premises$
- forward proof: $premises \xrightarrow{\tau} goals$

Classical Logic

example: The Hilbert System

Set of axioms:

$$A \rightarrow (B \rightarrow A) \quad (\text{A1})$$

$$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \quad (\text{A2})$$

$$A \vee \neg A \quad (\text{EM})$$

Single inference rule (*Modus Ponens*)

$$\llbracket A, A \rightarrow B \rrbracket \longrightarrow B \quad (\text{MP})$$

Some provable tautologies:

$$\neg\neg(A \vee \neg A) \quad (\text{nnEM})$$

$$A \rightarrow \neg\neg A \quad (\text{DNi})$$

$$\neg\neg A \rightarrow A \quad (\text{DNe})$$

$$((A \rightarrow B) \rightarrow A) \rightarrow B \quad (\text{PL})$$

$$\neg(A \wedge B) \rightarrow \neg A \vee \neg B \quad (\text{DMdi})$$

$$\neg(A \vee B) \rightarrow \neg A \wedge \neg B \quad (\text{DMci})$$

$$\neg A \wedge \neg B \rightarrow \neg(A \vee B) \quad (\text{DMce})$$

$$\neg A \vee \neg B \rightarrow \neg(A \wedge B) \quad (\text{DMde})$$

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Intuitionistic Logic

a.k.a. Constructive Logic

Set of axioms:

$$A \rightarrow (B \rightarrow A) \quad (A1)$$

$$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \quad (A2)$$

~~$$A \vee \neg A \quad (EM)$$~~

Single inference rule (*Modus Ponens*)

$$\llbracket A, A \rightarrow B \rrbracket \longrightarrow B \quad (MP)$$

Some provable tautologies:

$$\neg\neg(A \vee \neg A) \quad (nnEM)$$

$$A \rightarrow \neg\neg A \quad (DNi)$$

~~$$\neg\neg A \rightarrow A \quad (DNe)$$~~

~~$$((A \rightarrow B) \rightarrow A) \rightarrow B \quad (PL)$$~~

~~$$\neg(A \wedge B) \rightarrow \neg A \vee \neg B \quad (DMdi)$$~~

$$\neg(A \vee B) \rightarrow \neg A \wedge \neg B \quad (DMci)$$

$$\neg A \wedge \neg B \rightarrow \neg(A \vee B) \quad (DMce)$$

$$\neg A \vee \neg B \rightarrow \neg(A \wedge B) \quad (DMde)$$

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

First acquaintance



- based on **classical** higher-order logic
- created in 1986 at University of Cambridge and Technische Universität München
- highly automated
- uses functional language HOL
- has large collection of formalised theories



- based on **intuitionistic** logic (Calculus of Inductive Constructions)
- created in 1984 at INRIA (Paris, France)
- highly automated
- uses functional language Gallina
- has large collection of formalised theories

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of the basic datatypes



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of the basic datatypes



```
datatype bool =  
  True | False
```

```
datatype nat =  
  zero ("0") | Suc nat
```



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of the basic datatypes



```
datatype bool =  
  True | False
```

```
datatype nat =  
  zero ("0") | Suc nat
```



```
Inductive False : Prop := .
```

```
Inductive True : Prop := I : True.
```

```
Inductive nat : Type :=  
  | 0 : nat  
  | S : nat -> nat.
```

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of a recursive function



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of a recursive function



```
fun add ::  
  "nat  $\Rightarrow$  nat  $\Rightarrow$  nat"  
where  
  "add 0 n = n"  
| "add (Suc m) n =  
  Suc (add m n) "
```



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Definition of a recursive function



```
fun add ::  
  "nat  $\Rightarrow$  nat  $\Rightarrow$  nat"  
where  
  "add 0 n = n"  
| "add (Suc m) n =  
  Suc (add m n) "
```



```
Fixpoint add (n m: nat) : nat :=  
  match n with  
  | 0 => m  
  | S n' => S (n' + m)  
end
```

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Simple proof



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Simple proof



```
lemma de_morgan:
  " ( $\neg P \vee \neg Q$ )  $\implies \neg (P \wedge Q)$  "
apply (rule notI)
apply (erule conjE)
apply (erule disjE)
apply
  (erule notE, assumption)+
done
```



Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle & Coq

Simple proof



```
lemma de_morgan:
  " ( $\neg P \vee \neg Q$ )  $\implies \neg(P \wedge Q)$  "
  apply (rule notI)
  apply (erule conjE)
  apply (erule disjE)
  apply
    (erule notE, assumption)+
  done
```



```
Theorem de_morgan:
  forall P Q : Prop,
    ( $\neg P \wedge \neg Q$ )  $\rightarrow \neg(P \vee Q)$  .
Proof.
  intros P Q H [Hp Hq].
  destruct H as [Hnp | Hnq].
  - apply Hnp. assumption.
  - apply Hnq. assumption.
Qed.
```

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

Isabelle: Backward proof example

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

```
fun range_sum :: "nat => nat"
  where "range_sum n = ( $\sum$  k::nat=0..n . k)"

theorem arith_progr_sum: "2 * (range_sum n) = n * (n + 1)"
  proof (induct n)
    show "2 * range_sum 0 = 0 * (0 + 1)" by simp
  next
    fix n have "2 * range_sum (n + 1) = 2 * (range_sum n) + 2 * (n + 1)" by simp
    also assume "2 * (range_sum n) = n * (n + 1)"
    also have "... + 2 * (n + 1) = (n + 1) * (n + 2)" by simp
    finally show "2 * (range_sum (Suc n)) = (Suc n) * (Suc n + 1)" by simp
  qed
```

Isabelle: Proof process

The screenshot shows the Isabelle2017 Scratch.thy (modified) window. The left pane displays the source code of the theorem and its proof. The right pane shows the proof state, including the goal and the current subgoal.

Source Code (Scratch.thy):

```
theory Scratch
  imports Main
begin

fun range_sum :: "nat => nat"
  where "range_sum n = (∑ k::nat=0..n . k)"

theorem arith_progr_sum: "2 * (range_sum n) = n * (n + 1)"
proof (induct n)
  show "2 * range_sum 0 = 0 * (0 + 1)" by simp
next
  fix n have "2 * range_sum (n + 1) = 2 * (range_sum n) + 2 * (n + 1)" by simp
  also assume "2 * (range_sum n) = n * (n + 1)"
  also have "... + 2 * (n + 1) = (n + 1) * (n + 2)" by simp
  finally show "2 * (range_sum (Suc n)) = (Suc n) * (Suc n + 1)" by simp
qed
```

Proof State (Right Pane):

```
proof (prove)
goal (1 subgoal):
  1. n * (n + 1) + 2 * (n + 1) = (n + 1) * (n + 2)
```

The status bar at the bottom indicates the version (14.15), session ID (387/507), and the current file (isabelle.isabelle,UTF-8-Isabelle) with a timestamp of 202/1242MB 2:18 AM.

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system
Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance
Basic syntax
Example of proof

Summary

Coq: Proof process

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

The screenshot shows the CoqIDE interface with a file named `*scratch*`. The left pane contains the following Coq code:

```
Require Import Coq.omega.Omega.
Require Coq.Logic.Classical.

Fixpoint range_sum (n: nat) : nat :=
  match n with
  | 0 => 0
  | S p => range_sum p + (S p)
  end.

Lemma range_sum_lemma: forall n: nat,
  range_sum (n + 1) = range_sum n + (n + 1).
Proof.
  intros. induction n.
  - simpl; reflexivity.
  - simpl; omega.
Qed.

Theorem SimpleArithProgressionSumFormula_Coq:
  forall n, 2 * range_sum n = n * (n + 1).
Proof.
  intros.
  induction n.
  (* goal: '2 * range_sum 0 = 0 * (0 + 1)' *)
  - simpl; reflexivity.
  (* goal: '2 * range_sum (S n) = S n * (S n + 1)' *)
  - rewrite -> Nat.mul_add_distr_l.
    rewrite -> Nat.mul_1_r.
    rewrite -> (Nat.mul_succ_l n).
    rewrite <- (Nat.add_1_r n).
    rewrite -> range_sum_lemma.
    omega.
Qed...
```

The right pane shows a subgoal:

```
1 subgoal
n : nat
IHn : 2 * range_sum n = n * (n + 1)
----- (1/1)
2 * range_sum (S n) = S n * S n + S n * 1
```

The bottom status bar indicates: `Ready, proving SimpleArithProgressionSumFormula_Coq`, `Line: 26 Char: 38`, `Coq is ready`, and `0 / 0`.

Coq: Proof process

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

```
Require Import Coq.omega.Omega.
Require Coq.Logic.Classical.

Fixpoint range_sum (n: nat) : nat :=
  match n with
  | 0 => 0
  | S p => range_sum p + (S p)
  end.

Lemma range_sum_lemma: forall n: nat,
  range_sum (n + 1) = range_sum n + (n + 1).
Proof.
  intros. induction n.
  - simpl; reflexivity.
  - simpl; omega.
Qed.

Theorem SimpleArithProgressionSumFormula_Coq:
  forall n, 2 * range_sum n = n * (n + 1).
Proof.
  intros.
  induction n.
  (* goal: '2 * range_sum 0 = 0 * (0 + 1)' *)
  - simpl; reflexivity.
  (* goal: '2 * range_sum (S n) = S n * (S n + 1)' *)
  - rewrite -> Nat.mul_add_distr_l.
    rewrite -> Nat.mul_1_r.
    rewrite -> (Nat.mul_succ_l n).
    rewrite <- (Nat.add_1_r n).
    rewrite -> range_sum_lemma.
    omega.
Qed.
```

Messages Errors Jobs

SimpleArithProgressionSumFormula_Coq is defined

Ready Line: 32 Char: 5 Coq is ready 0 / 0

Coq: Proof process + verified code extraction

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary

The screenshot shows the CoqIDE interface with a menu bar (File, Edit, View, Navigation, Try Tactics, Templates, Queries, Tools, Compile, Windows, Help) and a toolbar. The main editor displays Coq code for a function `range_sum`. The code is split into two panels: the left panel shows the Coq definition, and the right panel shows the extracted code in Haskell and OCaml. The Coq code defines `range_sum` as a function from `Nat` to `Nat`, using a `case` expression to handle the base case (`0`) and the inductive case (`S p`). The extracted Haskell code defines `range_sum` as a function from `Nat` to `Nat`, using a `case` expression to handle the base case (`0`) and the inductive case (`S p`). The extracted OCaml code defines `range_sum` as a function from `nat` to `nat`, using a `let rec` expression to handle the base case (`0`) and the inductive case (`S p`). The Coq code also includes a `Qed.` statement at the end.

```
Require Import Coq.omega.Omega.
Require Coq.Logic.Classical.

range_sum :: Nat -> Nat
range_sum n =
  case n of {
    0 -> 0;
    S p -> add (range_sum p) (S p)}
```

```
(** val range_sum : nat -> nat **)

let rec range_sum = function
  | 0 -> 0
  | S p -> add (range_sum p) (S p)
```

Ready Line: 32 Char: 5 Coq is ready 0 / 0

Figure 3. Extracted function in Haskell

Figure 4. Extracted function in OCaml

Summary

- Two widespread theorem provers were considered: Isabelle and Coq
- The tools are based on different logics \Rightarrow
 - unprovable statements and invalid classical proofs in Coq
 - sometimes more complex proof in Coq
 - ★ *constructive* proof in Coq
- Nonetheless, they both may be used to solve applied problems, such as software testing and verification

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A formal system

Classical and Intuitionistic
logics

Isabelle & Coq

First acquaintance

Basic syntax

Example of proof

Summary