

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

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Outline

Foundations of Formal Approach

A Formal System

Classical and Intuitionistic Logics

Two Theorem Provers

Isabelle

Coq

Comparison of the Theorem Provers

Comparison

Examples of Proofs

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Elements of a Formal System

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

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

- ▶ 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

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

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 each ψ_i is either an axiom ϕ_{A_i} , or a formula inferred from the set of previously derived formulas according the rules of inference.

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

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

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Intuitionistic Logic

a.k.a. Constructive Logic

Set of axioms:

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

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

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

Single inference rule (*Modus Ponens*)

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

Some provable tautologies:

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

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

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

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

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

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

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

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

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Curry-Howard Correspondence

The direct relationship between computer programs and mathematical proofs.
[To be done]

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Isabelle: First Acquaintance

- ▶ a generic proof assistant
- ▶ based on classical higher-order logic
- ▶ created in 1986 by
 - ▶ Larry Paulson @ University of Cambridge, and
 - ▶ Tobias Nipkow @ Technische Universität München
- ▶ uses powerful functional language HOL
- ▶ the proof system core *Isabelle* is extended by various theories: Isabelle/HOL, Isabelle/ZF, Isabelle/CCL, etc.

Example 1: Definition of basic datatypes

```
datatype bool =  
  True | False  
  
datatype nat =  
  zero ("0") | Suc nat
```

Example 2: Definition of addition over nat

```
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

Two Theorem
Provers

Isabelle
Coq

Comparison of the
Theorem Provers

Comparison
Examples of Proofs

Summary

Coq: First Acquaintance

- ▶ a formal proof management system
- ▶ based intuitionistic logic (Calculus of Inductive Constructions)
- ▶ created at INRIA (Paris, France) in 1984
- ▶ uses powerful functional language Gallina
- ▶ has large collection of formalised theories
- ▶ widely used in software verification (proof code extraction)

Example 3: Definition of basic datatypes

```
Inductive False : Prop := .

Inductive True : Prop := I : True.

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

Example 4: Definition of addition over nat

```
Fixpoint add (n m: nat) : nat :=
  match n with
  | O => m
  | S n' => S (n' + m)
  end
where "n + m" :=
  (add n m) : nat_scope.
```

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Comparison

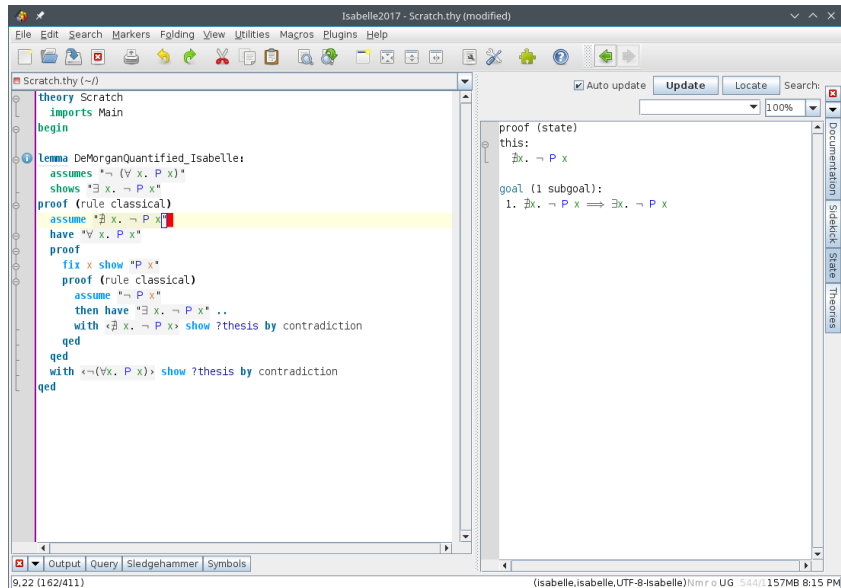
Major similarities:

- ▶ both work in a similar way of *verifying* the proof or *assisting* in creation of the new one
- ▶ $\text{premises} \xrightarrow{\text{tactics}} \text{goals}$ (forward proof)
- ▶ $\text{goals} \xrightarrow{\text{tactics}} \text{premises}$ (backward proof)
- ▶ both have large amount of libraries with formalised theories
- ▶ both dispose the set of highly automated tactics
- ▶ both are being actively developed these days

Major differences:

- ▶ based on different logics \Rightarrow
 - ★ unprovable statements and invalid proofs in Coq
 - ★ sometimes more complex proof in Coq
 - ★ *constructive* proof in Coq

Isabelle: Proof Example in Propositional Logic



The screenshot shows the Isabelle2017 IDE with the file Scratch.thy (modified). The interface includes a menu bar (File, Edit, Search, Markers, Folding, View, Utilities, Magros, Plugins, Help), a toolbar with icons for file operations and editing, and a status bar at the bottom showing the version 9.22 (162/411) and the current file path (isabelle,isabelle,UTF-8-Isabelle)Nmr o UG 544/1157MB 8:15 PM.

The main editor displays the following code:

```
theory Scratch
  imports Main
begin

lemma DeMorganQuantified_Isabelle:
  assumes "¬ (∀ x. P x)"
  shows "∃ x. ¬ P x"
proof (rule classical)
  assume "∃ x. ¬ P x"
  have "∀ x. P x"
  proof
    fix x show "P x"
    proof (rule classical)
      assume "¬ P x"
      then have "∃ x. ¬ P x" ..
      with <∃ x. ¬ P x> show ?thesis by contradiction
    qed
  qed
  with <¬ (∀ x. P x)> show ?thesis by contradiction
qed
```

The right-hand pane shows the proof state:

```
proof (state)
this:
  #x. ¬ P x

goal (1 subgoal):
  1. #x. ¬ P x ⇒ ∃ x. ¬ P x
```

The bottom status bar indicates the current position in the document: 9.22 (162/411).

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Isabelle: Proof Example over nat

The screenshot displays the Isabelle2017 IDE interface. The main window shows a theory named 'Scratch' with the following code:

```
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
```

The right-hand pane shows the proof state for the theorem 'arith_progr_sum'. It displays the goal (1 subgoal):

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

The bottom status bar shows the file path '14,15 (387/507)' and the system information '(Isabelle, Isabelle, UTF-8-Isabelle) Nm r o UG 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

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Coq: Proof Example in Propositional Logic

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 with icons for file operations, navigation, and execution. The main editor displays a Coq script for proving De Morgan's law. The script defines a theorem, states it for all propositions P and Q, and provides a proof using tactics like 'intros', 'split', 'destruct', and 'apply'. The right-hand pane shows the current subgoal and hypotheses. The bottom status bar indicates the proof is ready and shows line and character counts.

```
Theorem DeMorganPropositional_Coq:
  forall P Q : Prop, ~(P \/ Q) <-> ~P /\ ~Q.
Proof.
  (* 'tauto' automatically proves the equation *)
  intros P Q. unfold iff.
  split.
  - intros H_not_or. unfold not. constructor.
    + intro H_P. apply H_not_or. left. apply H_P.
    + intro H_Q. apply H_not_or. right. apply H_Q.
  - intros H_and_not H_or.
    destruct H_and_not as [H_not_P H_not_Q].
    destruct H_or as [H_P | H_Q].
    + apply H_not_P. assumption.
    + apply H_not_Q. assumption.
Qed.
```

Subgoal and Hypotheses:

```
1 subgoal
P, Q : Prop
H_not_or : ~ (P \/ Q)
H_Q : Q
----- (1/1)
P \/ Q
```

Messages | Errors | Jobs

Ready, proving DeMorganPropositional_Coq Line: 9 Char: 33 Coq is ready 0 / 0

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Coq: Proof Example over nat

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 a Coq script for proving a formula about the sum of an arithmetic progression. The script defines a function `range_sum`, a lemma `range_sum_lemma`, and a theorem `SimpleArithProgressionSumFormula_Coq`. The proof of the theorem is in progress, with the current goal being `2 * range_sum (S n) = S n * (S n + 1)`. The right-hand pane shows the current goal and the induction hypothesis (IHn). The bottom status bar indicates "Ready, proving SimpleArithProgressionSumFormula_Coq" and "Coq is ready".

```
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...
```

1 subgoal
n : nat
IHn : 2 * range_sum n = n * (n + 1)
$$\frac{}{2 * range_sum (S n) = S n * S n + S n * 1} (1/1)$$

Messages Errors Jobs

Ready, proving SimpleArithProgressionSumFormula_Coq Line: 26 Char: 38 Coq is ready 0 / 0

Comparison of two
theorem provers:
Isabelle & Coq

A. Yushkovskiy,
S. Tripakis

Foundations of
Formal Approach

A Formal System

Classical and Intuitionistic
Logics

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

Coq: Proof Example over nat + 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

Two Theorem
Provers

Isabelle

Coq

Comparison of the
Theorem Provers

Comparison

Examples of Proofs

Summary

The screenshot shows the CoqIDE interface with a Coq proof on the left and two extracted code snippets on the right. The Coq proof defines a function `range_sum` and proves a theorem `SimpleArithProgressionSumFormula_Coq`. The extracted code snippets show the function `range_sum` in Haskell and OCaml.

Figure 3. Extracted function in Haskell

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

Figure 4. Extracted function in OCaml

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

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

Ready, proving SimpleArithProgressionSumFormula_Coq Line: 26 Char: 38 Coq is ready 0 / 0

Summary

- ▶ Two widespread theorem provers were considered: Isabelle and Coq
- ▶ The **key difference** between them lie in differences between logical theories they based on
- ▶ 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

Two Theorem
Provers

Isabelle
Coq

Comparison of the
Theorem Provers

Comparison
Examples of Proofs

Summary