

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

Proof examples

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 Professor of Philosophy Columbia University
University of Virginia Emory 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 als „Disjunktionsprinzip“ auftretenden) Satzes vom ausgeschlossenen Dritten, daß entweder p oder q wenigstens 6 Elemente enthält.

Ebenso: Wenn man in der elementaren Arithmetik bewiesen hat, daß, wenn

Introduction

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

Proof examples

Summary

PRINCIPLES OF MATHEMATICAL LOGIC



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

Von L. E. J. Brouwer in Amsterdam.

§ 1.

von S
gesch
oder
wiese
Ausfu
bis z
liefert
fungen
bzw.
demz
mitus

licher
Dritte
oder
mente
Unm



wenigstens 11 Elemente enthält, so folgt hieraus auf Grund des (in diesem Falle
als „Disjunktionsprinzip“ auftretenden) Satzes vom ausgeschlossenen Dritten,
daß entweder p oder q wenigstens 6 Elemente enthält.

Ebenso: Wenn man in der elementaren Arithmetik bewiesen hat, daß, wenn
ein Zahlensystem S die Eigenschaften P und Q besitzt, dann

Outline

Foundations of Formal Approach

- A formal system

- Classical and Intuitionistic logics

Two Theorem Provers

- Isabelle & Coq

Comparison of the theorem provers

- Comparison

- Proof examples

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

Proof examples

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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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

Proof examples

Summary

Properties of a formal system

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

Proof examples

Summary

A formal system Γ is called:

- *consistent*, if $\nexists \phi \in \Gamma : \Gamma \vdash \phi \wedge \Gamma \vdash \neg \phi \Leftrightarrow \Gamma \not\vdash \perp$;
- *complete*, if $\forall \phi \in U : A \vdash \phi \vee A \vdash \neg \phi$;
- *independent*, if $\nexists a \in A : A \vdash a$.

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

Proof examples

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:

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

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

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

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

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

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

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

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

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

Proof examples

Summary

Isabelle & Coq

First acquaintance



- based on **classical** higher-order logic
- created in 1986 at University of Cambridge and Technische Universität München
- 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)
- 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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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 :=  
  | O : 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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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

Two Theorem
Provers

Isabelle & Coq

Comparison of the
theorem provers

Comparison

Proof examples

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  
  
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

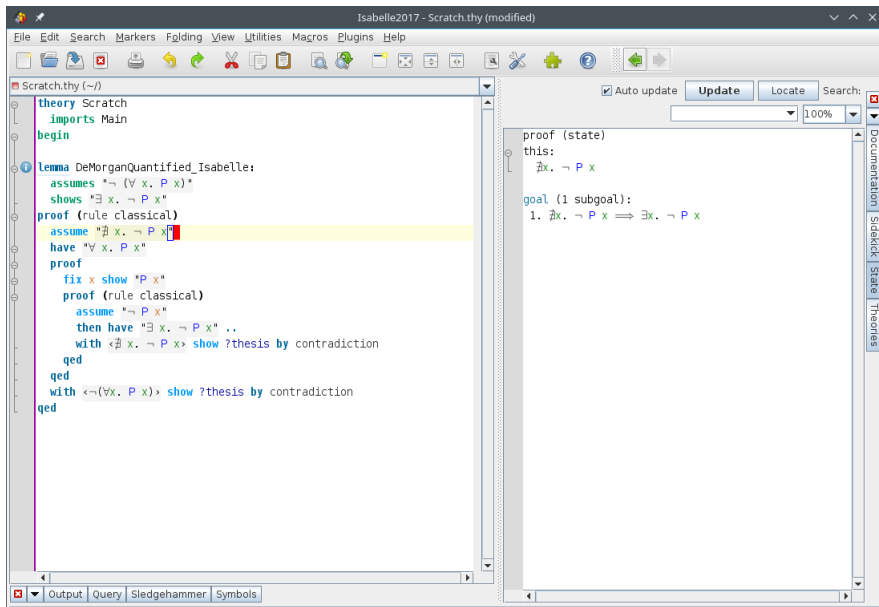
Comparison of the
theorem provers

Comparison

Proof examples

Summary

Isabelle: Native graphical user interface



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

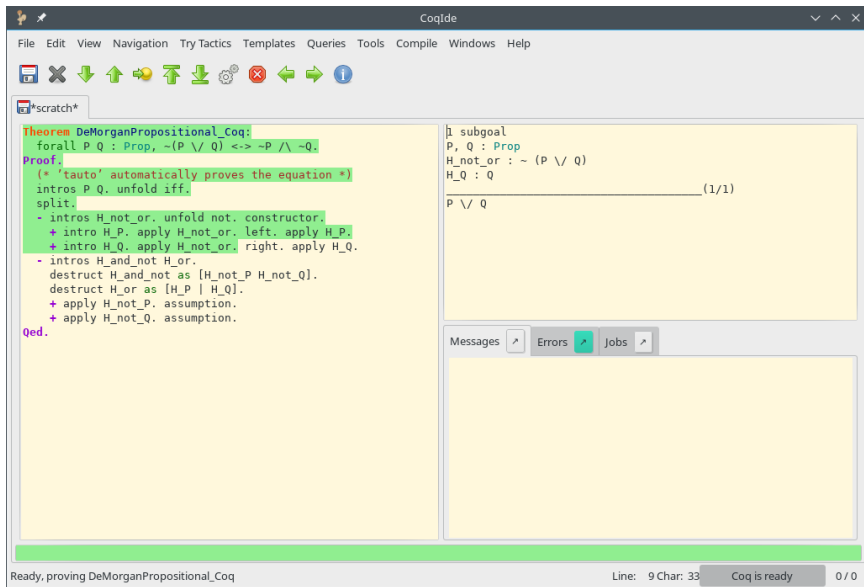
Isabelle & Coq

Comparison of the
theorem provers

Comparison
Proof examples

Summary

Coq: Native graphical user interface



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

Proof examples

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

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

Proof examples

Summary

Isabelle: proof example in propositional logic

The screenshot shows the Isabelle2017 IDE with a file named 'Scratch.thy (modified)'. The main editor displays a theorem named 'DeMorganQuantified_Isabelle' with its proof. The proof uses the 'rule classical' and 'show ?thesis by contradiction' tactics. The right-hand pane shows the current state of the proof, including the goal and the assumptions. The bottom status bar indicates the current position in the document: '9,22 (162/411)'.

```
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
```

proof (state)
this:
 $\exists x. \neg P x$

goal (1 subgoal):
1. $\exists x. \neg P x \implies \exists x. \neg P x$

9,22 (162/411) (isabelle,isabelle,UTF-8-Isabelle)Nm r o UG 544/157MB 8:15 PM

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

Proof examples

Summary

Isabelle: proof example over nat

The screenshot displays the Isabelle2017 IDE interface. The main window shows a file named `Scratch.thy` with the following content:

```
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 and the current proof steps:

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

The bottom status bar indicates the current position in the document: `14,15 (387/507)`. The bottom right corner shows the file path and size: `(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

Proof examples

Summary

Coq: proof example in propositional logic

The screenshot shows the CoqIDE interface with a file named `*scratch*`. The main editor contains a Coq proof script for De Morgan's law. The script is as follows:

```
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.
```

On the right side of the interface, a subgoal is displayed:

```
|1 subgoal
P, Q : Prop
H_not_or : ~ (P ∨ Q)
H_Q : Q
──────────────────────────────────────── (1/1)
P ∨ Q
```

At the bottom of the interface, the status bar indicates "Ready, proving DeMorganPropositional_Coq", "Line: 9 Char: 33", and "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

Proof examples

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 theorem about the sum of an arithmetic progression. The script defines a function `range_sum` and proves a lemma and a theorem. The right-hand pane shows the current subgoal and the proof steps taken so far. The bottom status bar indicates the current line and character position, and the Coq engine's status.

```
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)
$$\frac{2 * \text{range_sum } (S\ n) = S\ n * S\ n + S\ n * 1}{(1/1)}$$

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

Proof examples

Summary

Coq: proof example over nat + verified code

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

Proof examples

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` and its proof. A yellow box highlights the extracted code for both 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)
```

The Coq code in the background includes:

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

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

Lemma range_sum_
  range_sum (n
Proof.
  intros. indu
  - simpl; ref
  - simpl; ome
Qed.

Theorem SimpleAr
  forall n, 2
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...
```


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

Proof examples

Summary