Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A Foundational Formalization of Grounded Deduction in Isabelle/Pure

BSc Thesis

Sascha Kehrli
skehrli@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Prof. Dr. Bryan Ford
Prof. Dr. Roger Wattenhofer

05.07.2025

Abstract

Contents

Introduction 1

Background 2

2.1 Isabelle/Pure

Pure is the minimal meta-logic in Isabelle. The popular Isabelle/HOL fragment is formalized atop Pure, as are all other object logics in Isabelle. Pure is designed to be as generic as possible to allow implementing a wide range of object logics inside it. Aside from a bare-bones logic, Pure thus provides infrastructure to define types and axioms to facilitate the formalization of object logics in it.

Unfortunately, there is no single document that lays out the syntax, axioms, and derivation rules of Pure in its entirety. The following is an attempt at providing such a characterization of Pure, combining information from two Isabelle papers [1], [2] and the Isabelle reference manual [3].

2.1.1 Syntax of Pure

The core syntax of Pure is a typed lambda calculus, augmented with type variables, universal quantification, equality, and implication.

Propositions are terms of the distinct type `prop`. Propositions in Pure are thus terms and not types, like they are in type-theory based provers like Rocq or Lean.

Type Syntax

$$\begin{aligned}\tau ::= {} & \alpha \text{ (type variable)} \\ | {} & \tau \Rightarrow \tau \text{ (function type)} \\ | {} & \text{prop (type of propositions)}\end{aligned}$$

Term Syntax

$$\begin{aligned}t ::= {} & x \text{ (variable)} \\ | {} & c \text{ (constant)} \\ | {} & t\ t \text{ (application)} \\ | {} & \lambda x :: \tau . t \text{ (lambda abstraction)} \\ | {} & t \Longrightarrow t \text{ (implication)} \\ | {} & t \equiv t \text{ (equality)} \\ | {} & \bigwedge x :: \tau . t \text{ (universal quantification)}\end{aligned}$$

The symbols used for implication, equality, and universal quantification are non-standard to leave the standard symbols free for object logics atop Pure.

Even though Pure has type variables, it provides no construct to capture them as an argument, and thus also has no for-all type like the polymorphic lambda calculus System F.

2.1.2 Equality, Implication, and Quantification as Type Constructors

The connectives equality, implication, and universal quantification are all type constructors of the `prop` type with the following polymorphic type signatures.

$$\begin{aligned} \equiv &:: \alpha \Rightarrow \alpha \Rightarrow \text{prop} \\ \Longrightarrow &:: \text{prop} \Rightarrow \text{prop} \Rightarrow \text{prop} \\ \bigwedge &:: (\alpha \Rightarrow \text{prop}) \Rightarrow \text{prop} \end{aligned}$$

The arguments of $\equiv$ are the two operands to compare, the arguments for $\Longrightarrow$ are the sequent and consequent respectively, while the arguments of $\bigwedge$ are the type whose inhabitants are quantified over and the term that represents the body of the quantifier.

Since type variables denote only a single, albeit arbitrary, type, there is technically one instance of each connective for every given type. For example, for any type σ, there is a constant $\underset{\sigma}{\equiv} :: \sigma \Rightarrow \sigma \Rightarrow \text{prop}$.

2.1.3 Deduction Rules

The operational semantics of the underlying lamdba calculus and its typing rules are standard and thus omitted. The following discusses the more interesting deduction rules.

Relative to an object logic with a set of defined axioms Ω, any axiom $\omega \in \Omega$ can always be derived, as can any assumption $\gamma \in \Gamma$.

Basic Rules

$$\frac{A \text{ axiom}}{\Gamma \vdash A} \quad \text{(Axiom)} \qquad\qquad \frac{A \in \Gamma}{\Gamma \vdash A} \quad \text{(Ass)}$$

The implication and universal quantification introduction and elimination rules are standard.

Implication Deduction Rules

$$\frac{\Gamma \cup A \vdash B}{\Gamma \vdash A \Longrightarrow B} \quad (\Longrightarrow I) \qquad\qquad \frac{\Gamma_1 \vdash A \Longrightarrow B \quad \Gamma_2 \vdash A}{\Gamma_1 \cup \Gamma_2 \vdash B} \quad (\Longrightarrow E)$$

Universal Quantification Deduction Rules

$$\frac{\Gamma \vdash B(x) \quad x \text{ not free in } \Gamma}{\Gamma \vdash \bigwedge x.B(x)} \quad \left(\bigwedge I\right) \qquad\qquad \frac{\Gamma \vdash \bigwedge x.B(x)}{\Gamma \vdash B(a)} \quad \left(\bigwedge E\right)$$

For equality, besides the expected deduction rules corresponding to the equivalence relation properties, there are also deduction rules for equality of lambda abstractions and `prop`, the latter of which is defined as equivalence of truth values ($a \Leftrightarrow b$).

Equality Deduction Rules

$$\frac{}{\Gamma \vdash a \equiv a} \ (\equiv \text{Refl}) \qquad \frac{\Gamma \vdash b \equiv a}{\Gamma \vdash a \equiv b} \ (\equiv \text{Sym}) \qquad \frac{\Gamma \vdash a \equiv b \quad \Gamma \vdash b \equiv c}{\Gamma \vdash a \equiv c} \ (\equiv \text{Trans})$$

$$\frac{\Gamma \vdash a \equiv b}{\Gamma \vdash (\lambda x.a) \equiv (\lambda x.b)} \ (\equiv \text{Lam}) \qquad \frac{\Gamma \vdash a \Longrightarrow b \quad \Gamma \vdash b \Longrightarrow a}{\Gamma \vdash a \equiv b} \ (\equiv \text{Prop})$$

The λ-conversion rules facilitate equivalence reasoning for lambda abstractions. The rules are α-conversion, β-conversion and extensionality. The notation $a[y/x]$ expresses the substitution of x with y in a, that is, all occurences of x in a are replaced with y.

Lambda Conversion Rules

$$\frac{\text{y not free in } a}{\Gamma \vdash (\lambda x.a) \equiv (\lambda y.a[y/x])} \ (\alpha\text{-Conv}) \qquad \frac{}{\Gamma \vdash (\lambda x.a)\ b \equiv a[b/x]} \ (\beta\text{ -Conv})$$

$$\frac{\Gamma \vdash f\ x \equiv g\ x \quad \text{x not free in } \Gamma, f, \text{and } g}{\Gamma \vdash f \equiv g} \ (\text{Ext})$$

Finally, the equivalence elimination rule:

Equivalence Elimination

$$\frac{\Gamma \vdash a \equiv b \quad \Gamma \vdash a}{\Gamma \vdash b} \ (\equiv E)$$

2.1.4 Formalizing Object Logics in Pure

An object logic in Pure is created by adding new types, constants and axioms. That is, the Pure logic is extended.

2.2 Grounded Deduction (GD)

A References

[1] L. C. Paulson, “Isabelle: The Next 700 Theorem Provers.” [Online]. Available: https://arxiv.org/abs/cs/9301106

[2] L. C. Paulson, “The foundation of a generic theorem prover,” *J. Autom. Reason.*, vol. 5, no. 3, pp. 363–397, 1989, doi: 10.1007/BF00248324.

[3] L. Paulson, T. Nipkow, and M. Wenzel, “The Isabelle Reference Manual,” 1998.