363-364

ASTÉRISQUE

2015

SÉMINAIRE BOURBAKI
VOLUME 2013/2014
EXPOSÉ N° 1086

Thomas C. HALES

Developments in formal proofs

SOCIÉTÉ MATHÉMATIQUE DE FRANCE

Publié avec le concours du CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Astérisque est un périodique de la Société Mathématique de France.

Numéro 367-368

Diffusion

Maison de la SMF
Case 916 - Luminy
13288 Marseille Cedex 9
France
`smf@smf.univ-mrs.fr`

Hindustan Book Agency
O-131, The Shopping Mall
Arjun Marg, DLF Phase 1
Gurgaon 122002, Haryana
Inde

AMS
P.O. Box 6248
Providence RI 02940
USA
`www.ams.org`

Tarifs

Vente au numéro : 90 € ($ 135)

Abonnement Europe : 650 €, hors Europe : 689 € ($ 1 033)

Des conditions spéciales sont accordées aux membres de la SMF.

Secrétariat : Nathalie Christiaën

Astérisque
Société Mathématique de France
Institut Henri Poincaré, 11, rue Pierre et Marie Curie
75231 Paris Cedex 05, France
Tél : (33) 01 44 27 67 99 • Fax : (33) 01 40 46 90 96
`revues@smf.ens.fr` • `http://smf.emath.fr/`

ISSN 0303-1179

ISBN 978-2-85629-804-6

Directeur de la publication : Marc PEIGNÉ

Séminaire BOURBAKI June 2014
66e année, 2013-2014, n° 1086, p. 387 à 410

DEVELOPMENTS IN FORMAL PROOFS

by **Thomas C. HALES**

> *Si la mathématique formalisée était aussi simple que le jeu d'échecs, ... il n'y aurait plus qu'à rédiger nos démonstrations dans ce langage, comme l'auteur d'un traité d'échecs écrit dans sa notation ... Mais les choses sont loin d'être aussi faciles, et point n'est besoin d'une longue pratique pour s'apercevoir qu'un tel projet est absolument irréalisable.* — Bourbaki, 1966 [**16**, p. 5]

A proof assistant is interactive computer software that humans use to prepare scripts of mathematical proofs. These proof scripts can be parsed and verified directly from the fundamental rules of logic and the foundational axioms of mathematics. The technology underlying proof assistants and formal proofs has been under development for decades and grew out of efforts in the early twentieth century to place mathematics on solid foundations. Proof assistants have been built upon various mathematical foundations, including Zermelo-Fraenkel set theory (Mizar), Higher Order Logic (HOL), and dependent type theory (Coq) [**50, 36, 14**]. A *formal proof* is one that has been verified from first principles (generally by computer).

This report will focus on three particular technological advances. The HOL Light proof assistant will be used to illustrate the design of a highly reliable system. Today, proof assistants can verify large bodies of advanced mathematics; and as an example, we will turn to the formal proof in Coq of the Feit-Thompson Odd Order theorem in group theory. Finally, we will discuss advances in the automation of formal proofs, as implemented in proof assistants such as Mizar, Coq, Isabelle, and HOL Light.

1. BUILDING A TRUSTWORTHY SYSTEM WITH HOL LIGHT

HOL Light is a lightweight implementation of a foundational system based on Higher Order Logic (HOL). Because it is such a lightweight system, it is a natural system to use for explorations of the reliability of formal proof assistants.

1.1. Naive type theory

HOL, the foundational system of mathematics that we describe in this section, is based on a simply typed λ-calculus. This subsection describes a simple type theory in naive terms.

A salient feature of set theory is that it is so amorphous; everything is a set: ordered pairs are sets, elements of sets are sets, and functions between sets are sets. Thus, it is meaningful in set theory to ask bizarre questions such as whether a Turing machine is a minimal surface. In type theory, the very syntax of the language prohibits this question. Computer systems benefit from the extra structure provided by types.

Naively, a simple type system is a countable collection of disjoint nonempty sets called types. The collection of types satisfies a closure property: for every two types A and B, there is a further type, denoted $A \to B$, that can be identified with the set of functions from A to B.

In addition to types, there are terms, which are thought of as elements of types. Each term t has a unique type A. This relationship between a term and its type is denoted $t : A$. In particular, $f : A \to B$ denotes a term f of type $A \to B$.

There are variables that range over types called *type variables*, and another collection of variables that run over terms.

1.2. Models of HOL

The naive interpretation of types as sets can be made precise. We build a model of HOL in Zermelo-Fraenkel-Choice (ZFC) set theory to prove that HOL is consistent assuming that ZFC is. In this section, we review this routine exercise in model theory. At the same time, we will give some indications of the structure of HOL Light. See **[35]** for a more comprehensive introduction to HOL Light.

The interpretation of variable-free types as sets is recursively defined. We use a superscript M to mark the interpretation of a type as a set. Specifically, the types in HOL are generated by the boolean type `bool` (which we interpret as a set $\texttt{bool}^M = \{\top, \bot\}$ of cardinality two with labeled elements representing true and false) and the infinite type I (which we interpret as a countably infinite set I^M). Recursively, for any two variable-free types A and B, the type $A \to B$ is interpreted as the set $(A \to B)^M$ of all functions from A^M to B^M. We can arrange that the sets interpreting these types are all disjoint.

In summary so far, we fix an interpretation M, determining a countable collection $\mathcal{T} = \{A^M\}$ of nonempty sets in ZFC. We now extend our interpretation M to a *valuation* $v = (M, v_1, v_2)$, where v_1 is a function from the set of type variables in HOL to $\mathcal{T}$, and v_2 is a function from the set of term variables in HOL to $\bigcup \mathcal{T}$. The valuation v extends recursively to give a mapping that assigns a set $A^v \in \mathcal{T}$ to every

type A. We require v_2 to be chosen so that whenever x is a variable of type A, then $x^{v_2} \in A^v$. The valuation v extends recursively to give a mapping on all terms:

$$t \mapsto t^v \in A^v \in \mathcal{T}, \quad \text{for all } t : A.$$

For example, for every type A, there is a HOL term $(=)$ of type $A \to (A \to \texttt{bool})$ representing equality for that type.(1) This term is interpreted as the function in $(A \to (A \to \texttt{bool}))^v$ that maps $a \in A^v$ to the delta function δ_a supported at a (where the support of the function means the preimage of $\top$).

A *sequent* is a pair (L, t), traditionally written $L \vdash t$, where L is a finite set of terms called the *assumptions*, and t is a term called the *conclusion*. The terms of L and t must all have type `bool`. If L is empty, it is omitted from the notation.

If L is a finite set of Boolean terms, and if v is a valuation extending M, write L^v for the corresponding set of elements of the set $\texttt{bool}^M$. We say a sequent $L \vdash t$ is *logically valid* if for every valuation v for which every element of L^v is $\top \in \texttt{bool}^M$, we also have $t^v = \top$ in $\texttt{bool}^M$.

A *theorem* in HOL is a sequent that is generated from the mathematical axioms and rules of logic. There is a constant `FALSE` in HOL. The following amounts to saying that HOL does not prove `FALSE`.

THEOREM 1. — *If ZFC is consistent, then HOL is consistent.*

Proof sketch. — We give the proof in ZFC. Here, HOL is treated purely syntactically as a set of strings in a formal language.

We run through the rules of logic of HOL one by one and check that each one preserves validity.(2) For example, the reflexive law of equality in HOL states that for any term t of any type A, we have a theorem $\vdash t = t$. By the interpretation of equality described above, under any valuation v, this equation is interpreted as the value $\delta_{t^v}(t^v) \in \texttt{bool}^M$, which is $\top$. Hence the reflexive law preserves validity. The other rules (transitivity of equality, and so forth) are checked similarly.

We may well-order each set in the collection $\mathcal{T}$. HOL posits a choice operator of type $(A \to \texttt{bool}) \to A$ for every type A. The well-ordering allows us to interpret HOL's choice operator as an operator that maps a function $f \in (A \to \texttt{bool})^v$ with nonempty support to the minimal element of its support.

(1) The convention in HOL is to curry functions: using the bijection $X^{Y \times Z} = (X^Z)^Y$ to write a function whose domain is a product as a function of a single argument taking values in a function space. In particular, equality is a curried function of type $A \to (A \to \texttt{bool})$ rather than a relation on $A \times A$.

(2) There are ten such rules, giving the behavior of equality, λ-abstractions, β-reduction, and the discharge of assumptions. For reference purposes, an appendix lists the inference rules of HOL. The analysis in this section omits the rules for the creation of new term constants and types.