A Lost Proof

Christoph Benzmüller[1] and Manfred Kerber[2]

[1] Fachrichtung Informatik, Universität des Saarlandes
66041 Saarbrücken, Germany
chris@ags.uni-sb.de
[2] School of Computer Science
The University of Birmingham, Birmingham B15 2TT, England
M.Kerber@cs.bham.ac.uk

Abstract. We re-investigate a proof example presented by George Boolos which perspicuously illustrates Gödel's argument for the potentially drastic increase of proof-lengths in formal systems when carrying through the argument at too low a level. More concretely, restricting the order of the logic in which the proof is carried through to the order of the logic in which the problem is formulated in the first place can result in unmanageable long proofs, although there are short proofs in a logic of higher order.
Our motivation in this paper is of practical nature and its aim is to sketch the implications of this example to current technology in automated theorem proving, to point to related questions about the foundational character of type theory (without explicit comprehension axioms) for mathematics, and to work out some challenging aspects with regard to the automation of this proof – which, as we belief, nicely illustrates the discrepancy between the creativity and intuition required in mathematics and the limitations of state of the art theorem provers.

1 Introduction

The art of general purpose automated theorem proving has been best developed for first-order logic. State of the art systems like OTTER or SPASS are very powerful and can solve many problems of large collections of problems (see http://www.cs.jcu.edu.au/~tptp/TPTP). A particular highlight of the success of first-order theorem provers was the machine-generated proof of the Robbins problem [McC97]. While there are problems which are inherently of a higher-order nature and for which mere first-order mechanisms seem not to be appropriate (like diagonalisation proofs) surprisingly even for first-order problems to stay in first-order logic for the proof search may not be a good idea in general. Boolos gives a first-order example (actually it is a parameterised class of examples) that clearly sketches the limitations of first-order logic. Since there are complete calculi for first-order logic there is a first-order proof for Boolos' problem, but the proof is so long that it is intractable in first-order logic. Actually, it is easy to see that any bound on the proof length can be transgressed in

first-order logic, while in higher-order logic there is a short proof which can be summarised in one page.

Higher-order logic allows quantification over function and predicate variables and thus provides a far more expressive language than first-order logic. It was Bertrand Russell [Rus02,Rus03] who first pointed out in 1902 that in connection with the unrestricted comprehension principles[1] this may allow for *paradoxes*.[2] The most prominent example is the set of all non-self-containing sets (also called *Russell's paradox*). As a possible solution Russell suggested a few years later in [Rus08] a theory of types as a basis for the formalisation of mathematics that differentiates between objects and sets (or functions) consisting of these kinds of objects. This idea was also taken up by Alonzo Church in 1940, who introduced a formulation of type theory [Chu40], also called *simple theory of types*, based on Russell's ideas which is taken as a basis formalism in many modern higher-order theorem provers. Church's lambda calculus and his theory of types also opened the door for the propositions as types idea and constructive type theory.

TPS [ABB00,ABI+96], HOL [Gor85], PVS [ORS92], ΩMEGA [BCF+97], *λCLAM* [RSG98], LEO [BK98b] are interactive and automated higher-order provers based on the simple theory of types. Some prominent provers working for constructive type theory are NUPRL [CAB+86], COQ [CCF+95], and LEGO [LP92].

Automating proof search in higher-order logic is a very challenging enterprise, such that the above systems all provide facilities to combine interaction with automation. The idea is that the interactive human provides the crucial proof steps while simple subgoals are handled automatically by the prover. Of course, many non-trivial proofs can be already automated in higher-order logic and the most impressive record in this sense is given by the theorems proven automatically with TPS [ABB00,ABI+96]. A well known and very simple example illustrating the expressiveness and elegance of automated higher order theorem proving is Cantor's theorem, where the diagonalisation argument, in form of a λ-term, is synthesised by higher-order unification. For further and more impressive examples we refer to [ABI+96].

However, as we will discuss below Boolos' example perspicuously demonstrates the limitations of current first-order and higher-order theorem proving technology. As we will illustrate, with current technology it is not possible to find his proof automatically, even worse, automation seems very far out of reach. Let's first give a high-level description why this is so. Firstly, Boolos' proofs need comprehension principles to be available and it employs different complex instances of them. It is important to note here that theorem provers based on Church's theory of types typically assume that the comprehension principles can be completely avoided due to the power of higher-order unification and the existence of λ-terms. First-order theorem provers use no comprehension principle

[1] These principles assure the existence of certain functions, we will come back to them in Subsection 2.1.

[2] Paradoxes occur equally in first-order axiomatisations of set theory with unrestricted comprehension principles.

nor typically any other definition principle to introduce new concepts. Secondly, the particular instances of the comprehension axioms cannot be determined by higher-order unification but are so-called *Eureka*-steps which have to be *guessed*. However the required instantiations here are so complex that it is unrealistic to assume that they can be guessed, for instance, by blindly applying the primitive substitution principle [ABI+96], by splitting rules [Hue72,Hue73], or by concisely representing them so that new ones can be generated during the proof search [Ker94b]. Here it is where human intuition and creativity comes into play, and the question arises how this kind of creativity can be realised and mirrored in a theorem prover.

In the following we concentrate on the classical notion of higher-order logic and, in particular, on the simple theory of types, that is, a classical higher-order logic based on Church's simply typed λ-calculus [Chu40]. For an introduction to the pure simply typed λ-calculus we refer to [Bar84].

Even though our viewpoint here is from classical higher-order logic the problem is more general and applies to constructive type theory [ML84] and other approaches as well.

2 Automating Higher-Order Proof Search

The automation of the simple theory of types was pioneered by the resolution based methods discussed in [And71,Hue72,JP72]. The authors introduce formal derivation systems and discuss soundness and completeness with respect to Henkin semantics [Hen50], respectively general models [And72a,And72b]. Huet combined his constrained resolution approach with a higher-order pre-unification algorithm [Hue75] which avoids guessing aspects of full higher-order unification. More recent approaches are discussed in [Wol93,BK98a] and a comparison of these approaches is provided by [Ben01].

In the rest of this section, we briefly discuss some specifics of theorem proving in higher-order logic.

2.1 Comprehension

The λ-binding construct in combination with the λ-conversion rules in the simple theory of types have the effect that the type restricted *comprehension axioms* become derivable (see also [And86, Chapter 5, p. 159]). The comprehension axioms are instances of the following axiom schema

$$\exists N_{\overline{\alpha^n}\to\beta}.\forall \overline{z^n}.N(z^1,\ldots,z^n) = B_\beta$$

Here B stands for an arbitrary term such that the variable N does not occur free in it. The intention of the comprehension axioms is to guarantee for each expression B the existence of the functions (or sets) referred by $N_{\overline{\alpha^n}\to\beta}$. For instance, if β is the type o of truth values the comprehension principle asserts that for any respective formula B there exists a set N (referred to by the variable N) which

contains exactly all those n-tuples for which B evaluates to true. The strength of the λ-notation lies in the fact that the required sets and functions N can easily be directly described by the term $\lambda\overline{z^n}.B$, so that the implicit requirement that each term in the simple theory of types has a denotation, already ensures their existence.

The theorem proving approaches from above as well as the systems mentioned in the introduction therefore avoid the infinitely many comprehension axioms altogether.

2.2 Primitive Substitution

In higher-order proof search some set variables may have to be instantiated with complex terms which cannot be synthesised by higher-order unification. Simple examples illustrating this problem are $\exists x_o.x$ or $\exists p_{\iota\to o}.\forall y_\iota.\neg p(y)$. In a resolution approach, for instance, the second formula leads to an initial unit clause $p(Y)$, where p is a free set variable and Y a Skolem term for y. There is no further partner clause with a complementary literal available and thus no resolution step is possible. However, when instantiating the set variable p with the (complementary) set $\lambda x.\neg p(x)$ leading to a second clause $\neg p(Y)$, then a proof can be immediately constructed. Note that this proof in some sense mirrors an initial instantiation of set variable p by a term $\lambda x.p(x)\wedge\neg p(x)$ denoting the empty set. Unfortunately this set description cannot be synthesised by unification in the context. An example which requires more complex instantiations is X5310 from the TPS library, which is discussed in detail in [ABI+96, pp.333-335]. There, for instance, an instantiation $\lambda p_{\beta\to o}\lambda y_\beta.\exists x_\beta.p(x)\to p(y)$ is employed.

In Huet's resolution approach the *splitting rules* address this problem, which are replaced in Andrews' work [And89] by the more elegant *primitive substitution* principle. Splitting or primitive substitution blindly instantiates free set variables at literal head positions with a most general binding (partial binding) that imitates a logical connective, that is, with a most general formula that introduces a logical connective as head. For instance, the set of most general bindings for head variable p of the literal $p(Y)$ from above is $p \longleftarrow \lambda z.\neg b^1(z), p \longleftarrow \lambda z.b^2(z) \vee b^3(z), p \longleftarrow \lambda z.\forall w_\beta.b^4(z,w)$ where $b^{1,2,3}_{\iota\to o}$ and $b^4_{\iota\to(\beta\to o)}$ are new free head variables. Since there are infinitely many universal quantifiers in the simple theory of types (one for each type β) splitting and primitive substitution are infinitely branching. And as new free predicate variables are introduced they are recursively applicable to the results of their application. From an abstract point of view they realise a blind enumeration of all formula schemes in the Herbrand universe.

Clearly an unrestricted blind enumeration in this sense is practically infeasible. [ABI+96] therefore explains how this problem is handled in the TPS prover, which instead of a blind enumeration works with some selectively chosen instances.

2.3 Extensionality

Another source of blind search in higher-order which is in contrast to comprehension and primitive substitution less relevant for this paper is extensionality reasoning. The (type parameterised) functional extensionality principles

$$\forall m_{\alpha\to\beta}.\forall n_{\alpha\to\beta}.m = n \leftrightarrow \forall x.m(x) = m(x)$$

and Boolean extensionality principle

$$\forall p_o.\forall q_o.p = q \leftrightarrow (p \leftrightarrow q)$$

are both valid in Henkin semantics. In order to guarantee Henkin completeness these infinitely many axioms[3] are assumed to be available in the search space of nearly all approaches. The problem is that the $\beta\eta$-reasoning facilities built-in to higher-order unification are not sufficient to cover all extensionality aspects in theorem proving, such that blind search with these axioms is additionally required. In practical applications, however, the extensionality axioms are at the expense of completeness entirely or at least partly avoided. A complete approach which avoids these axioms in the search space and instead employs a goal directed extensionality treatment is provided by [BK98a].

3 "A Curious Inference"

In [Boo98, Chapter 25, p. 376–382] Boolos presents an example of a first-order problem which has only a very long derivation in first-order logic, but which has a short derivation in a second-order logic, by making use of comprehension axioms. He builds up on earlier work by Gödel [Göd36] who showed that in a higher-order logic by going to higher levels it may be possible to obtain short proofs.

In this section we take a closer look at this example and shed light on the question why a comprehension axiom of second-order logic is needed and even first-order with definition principle wouldn't suffice to generate short proofs.

Boolos assumes an inference system in the first-order predicate calculus with identity and function symbols s (unary) and f (binary).

1. $\forall n.f(n,1) = s(1)$
2. $\forall x.f(1,s(x)) = s(s(f(1,x)))$
3. $\forall n.\forall x.f(s(n),s(x)) = f(n,f(s(n),x))$
4. $D(1)$
5. $\forall x.(D(x) \to D(s(x)))$

   $\therefore$
6. $D(f(s(s(s(s(1))))),s(s(s(s(1))))))$

[3] Polymorphism may help to avoid infinitely many extensionality axioms in the search space. However, polymorphism does solve the search space problem induced by blind forward search with the extensionality principles.

Notice that there is no induction principle available. If it were available there would be a proof in two steps, namely because of (4) and (5), we would get $\forall x_{\blacksquare} D(x)$ by induction and hence $D(f(s(s(s(s(1))))), s(s(s(s(1))))))$ by $\forall$-elimination.

Actually, there is a relatively easy but enormously long proof. f is an Ackermann function and it is possible to prove the conclusion by $f(s(s(s(s(1))))), s(s(s(s(1))))) - 1$ many application of Modus ponens from (4) with (5) to come to (6). Function f as defined in (1)–(3) grows extremely fast and hence $f(s(s(s(s(1))))), s(s(s(s(1)))))$ is a very big number[4] such that there is no chance to actually perform so many applications with all computation power ever.

A proof in second-order logic makes use of comprehension axioms, in particular, of the following two[5]:

$$\exists N_{\blacksquare} \forall z_{\blacksquare} N(z) \leftrightarrow (\forall X_{\blacksquare} X(1) \wedge \forall y_{\blacksquare} (X(y) \rightarrow X(s(y))) \rightarrow X(z))$$

$$\exists E_{\blacksquare} \forall z_{\blacksquare} E(z) \leftrightarrow (N(z) \wedge D(z))$$

With this comprehension principle Boolos proves strong lemmata which can be viewed as conditional induction principle of the kind: "assumed the induction principle holds for number z – corresponding to $N(z)$ – then we can show for any predicate X a property $X(z)$ by induction."

The proof goes as follows; first you establish the following lemmata:

Lemma 1: $N(1)$,$\forall y_{\blacksquare}(N(y) \rightarrow N(s(y)))$, $N(s(s(s(s(1))))))$, $E(1)$, $\forall y_{\blacksquare}(E(y) \rightarrow E(s(y)))$, and $E(s(1))$

Proof sketch: These are easy consequences of the axioms.

Lemma 2: $\forall n_{\blacksquare} N(n) \rightarrow \forall x_{\blacksquare}(N(x) \rightarrow E(f(n,x)))$

Proof sketch:

- Define $M(n) \leftrightarrow (\forall x_{\blacksquare} N(x) \rightarrow E(f(n,x))$. We want $\forall n_{\blacksquare}(N(n) \rightarrow M(n))$. Enough to show $M(1)$ and $\forall n_{\blacksquare}(M(n) \rightarrow M(s(n)))$, since then from $N(n)$ follows $M(n)$ by definition of $N(n)$ as

  $$N(z) \leftrightarrow (\forall X_{\blacksquare} X(1) \wedge \forall y_{\blacksquare}(X(y) \rightarrow X(s(y))) \rightarrow X(z))$$

  We can instantiate X by M, in particular, the definition of N does not refer to M and is a proper definition.

[4] Number in the sense that $f(s(s(s(s(1))))), s(s(s(s(1)))))$ unravels into the successive application of a very big number of s's to 1; in fact, Boolos shows that this number is at least the result of an exponantial stack $2^{(2^{\cdots 2})}$ containing 64K '2s' in all.

[5] In the comprehension schema in Subsection 2.1 we employed $=$ to cover the general case. Here N is a predicate and $N(z)$ a term of type o such that $=$ can be replaced by $\leftrightarrow$ by boolean extensionality as introduced in Subsection 2.3.

The rest of the proof of the lemma is mainly a further reduction of the problem in a similar way, the theorem itself is an easy application of the two lemmata.

Boolos' proof makes a couple of times use of the comprehension principle:

Subgoal to prove	comprehension axiom applied
$\forall n. N(n) \to (\forall x. N(x) \to E(f(n,x)))$	$\exists M. \forall n. M(n) \leftrightarrow (\forall x. N(x) \to E(f(n,x)))$
$\forall x. N(x) \to E(f(1,x))$	$\exists Q. \forall x. Q(x) \leftrightarrow E(f(1,x))$
$\forall x. N(x) \to E(f(s(n),x))$ from $\forall x. N(x) \to E(f(n,x))$	$\exists P. \forall x. P(x) \leftrightarrow E(f(s(n),x))$

In plain first-order logic this cannot be modelled. Even in first-order logic *with* definition principle it can't. Namely if we try to do so we run into problems. For instance, we may try to define $N(n)$ as $M(1) \wedge \forall y. (M(y) \to M(s(y))) \to M(n)$, but this is no longer a proper definition, since now N is defined in terms of M and M in terms of N, that is, this higher-order construction cannot be modelled in first-order logic. The original definition of N heavily depends on the universal second-order quantifier $\forall X$, in which X can be later instantiated by predicates which are defined in terms of N itself.

Boolos' example – as well as Gödel's results – show that even for cases in which it is sufficient to stay completely in first-order logic *in principle*, it may not be sufficient *in practice*.

4 Implications, Questions, and Challenges

In this section we discuss the implications of this example for recent higher-order theorem proving approaches, formulate some related foundational questions, and explain why we consider Boolos' example as a challenge in automated reasoning for the new millennium.

4.1 Implications for existing Higher-order theorem proving approaches

For higher-order automated theorem proving there is the very difficult question how it may be determined whether certain comprehension axioms are necessary or not. Note that the comprehension axiom required in the above proof *can not* be replaced by the unification with a λ-expression, since Boolos' problem is initially a mere first-order problem, that is, there is no second-order term given to unify with. Since there is in particular no higher-order variable given in the problem formulation also primitive substitution is not applicable. Hence, without comprehension and higher-order variables there is no chance at all to introduce any kind of structures that could support Boolos proof.

Of course, theoretically there are the extensionality axioms which contain higher order variables. Furthermore, in our particular case one could think of adding some new axioms like

$$\forall b_o. \exists N_{\overline{\alpha^n} \to o}. \forall \overline{z^n}. N(\overline{z^n}) = b$$

instead of *all* comprehension axioms. The idea would be to leave the particular structure of the terms to the right open at the beginning and to employ primitive substitution to generate respective instances for b (which may depend on z) during proof search. A third option is to introduce free variables via the following tertium non datur axiom

$$\forall b_{o\blacksquare} b \vee \neg b$$

Whereas it seems to be impossible to derive the required comprehension axioms from the extensionality axioms by applying the primitive substitution principle, this is far more promising in the latter cases. However, as remarked in Subsection 2.2, without guidance, primitive substitution just subsequently enumerates formula schemes, such that this would differ only slightly from a direct enumeration of all comprehension axioms (for terms B of type o).

As remarked in Subsection 2.1 current approaches typically completely avoid the comprehension axioms (in practical applications the extensionality axioms and the tertium non datur axiom are avoided as well). Thus, Boolos elegant higher-order proof is not supported in these approaches. However, there exist completeness proofs for them ensuring that any formula that is valid in Henkin semantics can indeed be shown as such. Of course, the intractable first-order proof for Boolos' examples can theoretically be synthesised, so the example is not a counterexample to, but an example of these completeness result. It demonstrates well what completeness means, namely that practically a proof may not be found since proofs are prohibitively long. While this is clear for problems which do not have short proofs, Boolos' example has a practically tractable (with respect to its size) higher-order proof. It gets lost, however, in standard higher-order theorem proving because the comprehension schemes (or alternative axioms) are not available. Hence the question arises whether it makes sense at all to avoid the comprehension principles in the simple theory of types if this has the side effect that some tractable proofs get lost thereby. Especially in systems aiming at supporting non-trivial mathematics, where similar elegant proof constructions are potentially required this could be a serious problem.

4.2 Why is Boolos' example so challenging

Without any doubt the problem above is a challenge problem for automated theorem provers and we try to speculate in the following how it may be possible to solve such hard problems systematically. However, it should be very clear that these problems will remain very hard for the foreseeable future. While the problem to search in the original space is too hard, the problem to bring arbitrary comprehension axioms into play will open a different dimension with comparatively short proofs but extremely bushy search spaces, the only way to restrict that in some way seems at the moment to see how humans can master such search problems. Different observations can be made on that:

- Firstly, this problem cannot be solved by simply following the obvious path. In order to recognise that and to try to avoid the trap of following myriads of modus ponens rules, one would either have to know a priori that an

Ackermann function is involved and brute force won't lead to success, or after a while – on reflection of the progress – one would recognise that a brute force approach is not promising.
- Secondly, a human mathematician can recognise that the problem would be easy, if the induction principle were given. This again – as in the previous point – requires explicit knowledge of this principle. If somebody doesn't know the induction principle he/she has almost no chance to invent the trick.
- Thirdly, the insight that with induction the problem would be simple gives some chance to speculate the idea to prove the theorem in the form of assumed reasoning of the type: "*assume we had an induction principle for a particular number, then we could prove a particular statement. Since we don't have it, we assume it by a predicate N, which is defined by a comprehension axiom.*" This is certainly a very sophisticated reasoning step, and to our knowledge no system currently available is able to do it. The attempts to lemma speculation are all much more direct and the lemma speculated is the one that is needed to continue search.
- Fourthly, the way how the comprehension axioms are formed in the proof is quite systematic and they are closely related to the problem. It would be necessary to follow a similar line of systematic exploration.
- Fifthly the problem will remain difficult even if the questions above are answered, since on the one hand a new approach with defining new concepts has to be applied, and on the other hand it has to be applied not only once, but several times. This is a significant problem, since typically it is not so clear in an automated theorem prover whether it makes progress towards a solution. In some way in Boolos' proof it seems to be very clear that there is progress towards a proof, since the formulae to be shown in the sub-statements are getting smaller. Automated theorem provers would need an introspection capability and some better – more high-level – criterion for progress.

4.3 How to find a solution?

In this Subsection we speculate what techniques are around that can be explored to get a handle on finding Boolos' proof automatically.

- Firstly, it is clear that we need pretty high-level reasoning. Proof planning (as introduced by Bundy [Bun88]) is such a high-level form of reasoning. It would be necessary to build high-level methods which can address the type of problem we try to address.
- Proof planning links in with a different observation from above, namely that a lot of knowledge is necessary to attack the problem successfully. Unfortunately, most systems seem not to benefit from big amounts of data, but get confused, since they can't see the wood for the trees. A better way to structure knowledge bases seem to be an issue to address here.

- We already pointed to the importance of *reflection*. An experienced deductionist would certainly derive in an initial problem analysis the insight that monotonous modus ponens applications will not be successful here while an unexperienced person would reflect on the unsuccessful object level proof attempts in order to come up with a similar conclusion soon later. It thus could be sensible to develop theorem provers which not only construct proofs or proof plans at object level but in parallel also reflect on their object-level proof search in order to built up a kind of meta-level understanding for the problem at run time. Proof critics as discussed in [IB95] can be seen as a first step in this direction. Combined with an agent-based framework as discussed in [BJKS01] respective proof critical agents could work in parallel with the proof agents on object level.
- Another very important issue is to study re-representation in detail, that is, not to work with the problem as it is, but to restructure it first. Pólya gave the advise [Pól65, vol.2, p.80]: "*Of course you want to restate the problem (transform it into an equivalent problem) so that it becomes more familiar, more attractive, more accessible, more promising.*". McCarthy put up the mutilated checkerboard problem and its reformulations as a challenge problem [McC64].
- Selecting useful comprehension axioms is the same question as to form interesting concepts as done in Colton's work [CBW00]. For that reason it would be interesting to study how the work on concept formation can be integrated into a theorem proving framework.
- Human mathematicians are strongly guided in their proof search by models. It may be worthwhile to study how the selection process of suitable comprehension axioms can be guided by semantic, model-based techniques [Ker94a].

5 Conclusion

We re-investigated a very amenable proof example originally provided by Boolos to illustrate Gödels argument for the potentially drastic decrease of proof lengths when ascending to a logic of higher order. Aside from its intended illustrative character this example poses very practical questions and, as we belief, provides a good basis for a critical discussion of recent automated theorem proving technology. In particular we demonstrated that not only first-order theorem approaches are in principle condemned to fail when applied to this example but also recent higher-order approaches approaches and systems. The problem is that the avoidance of axioms like the comprehension principles (or alternative axioms which introduce free higher-order variables) in higher-order theorem proving obviously causes the loss of tractable proofs, like Boolos' elegant second-order proof for the given example. The expressiveness and power of higher-order logic is not employed to its full extend in recent higher-order approaches and on the given example they would operate as naïve as their first-order counterparts.

Therefore we think that Boolos example will remain a very challenging problem for automation for the foreseeable future. Because of its perspicuity it is

however well suited to speculate and investigate what kind of mechanisms (such as abstraction and reflection) will be required to fruitfully tackle it.

References

[ABB00] Peter B. Andrews, Matthew Bishop, and Chad E. Brown. System description: Tps: A theorem proving system for type theory. In David A. McAllester, editor, *Proceedings of 17th International Conference on Automated Deduction*, volume 1831 of *LNCS*, pages 164–169. Springer, June 2000.

[ABI+96] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A theorem proving system for classical type theory. *Journal of Automated Reasoning*, 16(3):321–353, 1996.

[And71] Peter B. Andrews. Resolution in type theory. *Journal of Symbolic Logic*, 3(36):414–432, 1971.

[And72a] Peter B. Andrews. General models and extensionality. *Journal of Symbolic Logic*, 37(2):395–397, 1972.

[And72b] Peter B. Andrews. General models, descriptions, and choice in type theory. *Journal of Symbolic Logic*, 37(2):385–394, 1972.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[And89] Peter B. Andrews. On Connections and Higher Order Logic. *Journal of Automated Reasoning*, 5:257–291, 1989.

[Bar84] Hendrik P. Barendregt. *The Lambda Calculus – Its Syntax and Semantics*. North Holland, 1984.

[BCF+97] C. Benzmüller, L. Cheikhrouhou, D. Fehrer, A. Fiedler, X. Huang, M. Kerber, M. Kohlhase, K. Konrad, E. Melis, A. Meier, W. Schaarschmidt, J. Siekmann, and V. Sorge. Ω Mega: Towards a mathematical assistant. In William McCune, editor, *Proceedings of the 14th Conference on Automated Deduction*, LNAI, Townsville, Australia, 1997. Springer Verlag.

[Ben01] C. Benzmüller. Comparing approaches to resolution based higher-order theorem proving. *Synthese, An International Journal for Epistemology, Methodology and Philosophy of Science*, 129, 2001. To appear.

[BJKS01] C. Benzmüller, Mateja Jamnik, Manfred Kerber, and Volker Sorge. An agent based approach to reasoning. KI-2001, 2001. To appear.

[BK98a] C. Benzmüller and M. Kohlhase. Extensional higher-order resolution. In Kirchner and Kirchner [KK98].

[BK98b] C. Benzmüller and M. Kohlhase. LEO – a higher-order theorem prover. In Kirchner and Kirchner [KK98].

[Boo98] George Boolos. *Logic, Logic, And Logic*. Harvard University Press, 1998.

[Bun88] Alan Bundy. The use of explicit plans to guide inductive proofs. In Ewing Lusk and Ross Overbeek, editors, *Proceedings of the 9th CADE*, pages 111–120, Argonne, Illinois, USA, 1988. Springer, LNCS 310.

[CAB+86] Robert L. Constable, S. Allen, H. Bromly, W. Cleaveland, J. Cremer, R. Harper, D. Howe, T. Knoblock, N. Mendler, P. Panangaden, J. Sasaki, and S. Smith. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[CBW00] Simon Colton, Alan Bundy, and Toby Walsh. Automatic invention of integer sequences. In Henry Kautz and Bruce Porter, editors, *AAAI-2000*, Austin, Texas, USA, 2000. AAAI Press, MIT Press.

[CCF+95] C. Cornes, J. Courant, J.-C. Fillitre, G. Huet, P. Manoury, C. Muoz, C. Murthy, C. Parent, Ch. Paulin-Mohring, A. Saïbi, and B. Werner. The Coq proof assistant reference manual, version 5.10. rapport technique 177, INRIA, July 1995.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[Göd36] Kurt Gödel. Über die Länge von Beweisen. *Ergebnisse eines mathematischen Kolloquiums*, **7**:23–24, 1936.

[Gor85] Mike Gordon. HOL: a machine oriented formulation of higher-order logic. Technical Report 68, University of Cambridge, Computer Laboratory, July 1985.

[Hen50] Leon Henkin. Completeness in the theory of types. *Journal of Symbolic Logic*, 15(2):81–91, 1950.

[Hue72] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.

[Hue73] Gérard P. Huet. A mechanization of type theory. In *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pages 139–146, 1973.

[Hue75] Gérard P. Huet. An unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[IB95] Andrew Ireland and Alan Bundy. Productive use of failure in inductive proof. *Special Issue of the Journal of Automated Reasoning*, 16(1–2):79–111, 1995.

[JP72] D. C. Jensen and Thomasz Pietrzykowski. A complete mechanization of (ω)-order type theory. In *Proceedings of the ACM annual Conference*, volume 1, pages 82–92, 1972.

[Ker94a] Manfred Kerber. How to use a first-order model generator for adjusting problem formulations of higher-order logic. In Ricardo Caferra, Chris Fermüller, Alex Leitsch, and Tanel Tammet, editors, *Proceedings of the CADE-Workshop on Automated Model Building*, pages 22–25, Nancy, France, 1994.

[Ker94b] Manfred Kerber. On the translation of higher-order problems into first-order logic. In Tony Cohn, editor, *Proceedings of the 11th ECAI*, pages 145–149, Amsterdam, The Netherlands, August 1994. John Wiley & Sons, Chichester, England.

[KK98] C. Kirchner and H. Kirchner, editors. *Proceedings of the 15th Conference on Automated Deduction*, number 1421 in LNAI, Lindau, Germany, 1998. Springer Verlag.

[LP92] Z. Luo and R. Pollack. Lego proof development system: User's manual. Technical report, Department of Computer Science, Edinburgh University, 1992.

[McC64] John McCarthy. A tough nut for proof procedures. AI Project Memo 16, Stanford University, Stanford, California, USA, 1964.

[McC97] William McCune. Solution of the Robbins problems. *Journal of Automated Reasoning*, **19**(3):263–276, 1997. see also `\tthttp://www.mcs.anl.gov/home/mccune/ar/robbins/.`

[ML84] Per Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

[ORS92] S. Owre, J. M. Rushby, and N. Shankar. PVS: a prototype verification system. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 748–752, Saratoga Spings, NY, USA, 1992. Springer Verlag.

[Pól65] George Pólya. *Mathematical Discovery – On Understanding, Learning, and Teaching Problem Solving.* Princeton University Press, Princeton, New Jersey, USA, 1962/1965.

[RSG98] Julian Richardson, Alan Smaill, and Ian Green. Proof planning in higher-order logic with λclam. In Kirchner and Kirchner [KK98], pages 129–133.

[Rus02] Bertrand Russell. Letter to Frege. Printed in [vH67], 1902.

[Rus03] Bertrand Russell. *The principles of mathematics.* Cambridge University Press, Cambridge, England, 1903.

[Rus08] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, XXX:222–262, 1908.

[vH67] Jean van Heijenoort. *From Frege to Gödel : a source book in mathematical logic 1879-1931.* Source books in the history of the sciences series. Harvard University Press, Cambridge, MA, USA, 3rd printing, 1997 edition, 1967.

[Wol93] David A. Wolfram. *The Clausal Theory of Types.* Cambridge University Press, 1993.