Towards an Interactive Mathematical Proof Language

Henk Barendregt

Abstract

Formalizing mathematical proofs has as aim to represent arbitrary mathematical notions and proofs on a computer in order to construct a database of certified results useful to learn and develop the subject. At present it is mathematically not appealing to construct formal proofs. To make formalzing more mathematician-friendly one should have a good interface for proofs, definitions and computations. The proof-assistant Mizar does have a good interface for proofs, but not for making computations. Other assistants, like Coq based on type theory, does have a good interface for computations, but not for proofs. This paper sketches ways in which proofs are represented in a mathematical way. Although the underlying formalized statements come from the system Coq, this is not essential. Mainly the paper has as aim to convince implementers of mathematical assistants to make systems in such a way that formalizing proofs becomes natural. Much further developed is the work on Isar providing a mathematical proof language for the assistant Isabelle. The approach in this paper is to approximate a proof language by writing *proof-sketches*, a notion by Wiedijk, with the aim that they should eventually be verifiable by a proof-checker. Nederpelt [2002] has a different approach: there the emphasis is on the ease of providing formalizations of mathematical definitions.

Acknowledgments. The author wishes to thank Freek Weedijk, for advocating the mathematical style of representing formal proofs in Mizar, and Michael Beeson for suggesting not to work towards constructing a new proof-assistant, but to *specify* one by providing a benchmark of candidate proof-scripts that should be accepted by the checker. Bas Spitters gave useful feedback and implemented part of the language.

1. Interactive proof-assistants

Mathematical assistants are workstations running a program that verifies the correctness of mathematical theorems, when provided with enough evidence. Systems for automated deduction require less evidence or even none at all; proof-checkers on the other hand require a fully formalized proof. We will focus on the problem of creating such *proof-objects* for mathematical statements for proof-checking.

In the pioneering Automath[1] system of Dick de Bruijn (see Nederpelt et al. [1994]), based on dependent type theory, proofs had to be given ready and well. The same applies to the system Mizar[2] (see Muzalewski [1993], Wiedijk [1999]), based on set-theory. On the other hand for systems like NuPrl[3] (see Constable et al. [1986]), Isabelle[4] (see Paulson [1994], Paulson [2002], Nipkow et al. [2002]), HOL[5] (see Gordon and Melham [1993], Harrison [2000]) and Coq[6] (see The Coq Development Team [2002]), the proofs are obtained in an interactive fashion between user and the proof-checker. Therefore one speaks about an interactive mathematical assistant. The list of statements that have to be given to such a checker, the *proof-script*, is usually not mathematical in nature, see e.g. section 3. The problem is that the script consists of fine-grained steps what should be done, devoid of any mathematical meaning.

Mizar is the only system having a substantial library of certified results in which the proof-script is mathematical in nature. Wiedijk speaks of the declarative style of Mizar. This paper is an attempt to provide an interactive script language for an interactive proof assistant like Coq, that is declarative and hence mathematical in flavor. A similar approach is found in the project Isar[7] (see Wenzel [1999], Wenzel [2002a], Wenzel [2002b]), with an implemented system.

In de Bruijn [1994] a plea was given to use a mathematical vernacular for formalizing proofs. This paper sketches a language MPL (*Mathematical Proof Language*) between informal mathematics and formalized mathematics with the claim that it can be translated automatically into the formalized

[1] <www.cs.kun.nl/~freek/aut>
[2] <www.mizar.org>
[3] <www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html>
[4] <www.cl.cam.ac.uk/Research/HVG/Isabelle>
[5] <archive.comlab.ox.ac.uk/formal-methods/hol.html>
[6] <pauillac.inria.fr/coq>
[7] <isabelle.in.tum.de/Isar>

language of interactive proof-assistants.

Two case studies are presented: Euclidean division with remainder and Newman's lemma. We present for these a mathematical proof "best style", a proof-sketch in the sense of Wiedijk (semi-formal proof that can be made formal easily) and a formal proof in Coq with mathematical mode.

2. Euclidean division with remainder

This small case study is about Euclidean division with remainder for the natural numbers. The result states that for every n and $d > 0$ there are q, r such that $n = dq + r$ and $r < d$. The proof uses cut-off subtraction $a \dot{-} b$, which is $a - b$ if $b \leq a$ and is 0 otherwise, and course of value induction.

2.1. Best mathematical style

2.1. Lemma. *The following results hold.*

(i) *subtr_sml* $\forall a, b{\in}\mathbb{N}[0 < a, b \;\Rightarrow\; a \dot{-} b < a]$.
(ii) *subtr_plus* $\forall a, b{\in}\mathbb{N}[a \geq b \;\Rightarrow\; (a \dot{-} b) + b = a]$.
(iii) *less_suc* $\forall n, m{\in}\mathbb{N}[n < (m+1) \;\Leftrightarrow\; (n < m \vee n = m)]$.

2.2. Proposition (Course of value induction). *Suppose that for all* $n{\in}\mathbb{N}$

$$(\forall k{<}n.P(k)) \;\Rightarrow\; P(n).$$

Then $\forall n{\in}\mathbb{N}.P(n)$.

Proof. Write

$$\text{before}(n, P) \;\Leftrightarrow\; \forall k{<}n.P(k).$$

Now assume

$$\forall n[\text{before}(n, P) \;\Rightarrow\; P(n)] \qquad (+)$$

in order to show $\forall n P(n)$. We will do this by showing by (ordinary) induction

$$\forall n.\text{before}(n, P),$$

because then for all $n{\in}\mathbb{N}$ one also has before$(n{+}1, P)$, in particular $P(n)$.

Basis $n = 0$. Indeed, before$(0, P)$ holds vacuously.

Induction step. Suppose before(n, P). Then by $(+)$ one has $P(n)$, hence $[\forall k{<}n.P(k)]$ & $P(n)$, therefore $\forall k{<}n{+}1.P(k)$, i.e. before$(n{+}1, P)$. ■

2.3. PROPOSITION (Division with remainder). *Let* $d{\in}\mathbb{N}$ *with* $0 < d$. *Then*

$$\forall n{\in}\mathbb{N}\exists q, r[r < d \,\&\, n = qd + r].$$

PROOF. Let $d{\in}\mathbb{N}$ with $0 < d$ be given. Write

$$P(n) := \exists q, r.[r < d \,\&\, n = qd + r].$$

We will show $\forall n{\in}\mathbb{N}.P(n)$ by course of value induction. So assume

$$\forall k < n.P(k), \tag{ih}$$

in order to prove $P(n)$. If $n < d$, then we can take $q = 0, r = n$. If on the other hand $n \geq d$, define $n' = n \dot{-} d$. Then $n' < n$ by subtr_sml. Therefore $P(n')$ by (ih). Hence for some q', r'

$$r' < d \,\&\, n' = dq' + r'. \tag{1}$$

Take $q = q' + 1, r = r'$. Then $r < d$ and

$$\begin{array}{rcll} n & = & (n \dot{-} d) + d, & \text{by subtr\_plus,} \\ & = & n' + d & \\ & = & (dq' + r') + d, & \text{by (1),} \\ & = & d(q' + 1) + r', & \text{by computation,} \\ & = & dq + r. \blacksquare & \end{array}$$

2.2. Proof sketch

```
Proposition (Division with remainder).
Let d:nat with 0<d.  Then for all n:nat
there exists q,r:nat such that
r<d &  n=d*q+r.

Proof.  Let d:N. Assume 0<d, towards
(n:N)(EX q:N)(EX r:N)[r<d & n=d*q+r].
Write P(n):=(EX q:N)(EX r:N)[r<d & n=d*q+r].
Apply course of value induction to P:
let n:N, assume
     (before n P)                                          (ih)
```

```
in order to show P(n).  Remember before in ih.
We have [n<d ∨ n>=d].  Remember P.
   Case n<d.  Take q=0, r=n.
Then r<d and we have n=d*0+n, by computing.  Done.
   Case n>=d.  Write nn:= n--d.  Then nn<n, by subtr_sml.
Therefore by ih
     P(nn)                                                    (H1)
Remember P in H1.  Hence for some qq and rr
      rr<d &nn=d*qq+rr.                                       (H2)
Take
     q=qq+1, r=rr.                                            (H3)
We have r<d and

        n = (n--d)+d,     by subtr_plus,
          = (d*qq+rr)+d,  by H2,
          = d*(qq+1)+rr,  by computation,
          = d*q+r,        by (H3).  QED
```

3. Newman's Lemma

As a second case study we specify for Newman's Lemma a feasible interactive mathematical proof development.

3.1. Best mathematical style

Again we start with the informal statement and proof.

Let A be a set and let R be a binary relation on A. R^+ is the transitive closure of R and R^* is the transitive reflexive closure of R.

The following properties of R are introduced: $\mathrm{CR}(R)$ the *Church-Rosser property*, $\mathrm{cr}(R,a)$ the *local CR property* and $\mathrm{WCR}(R)$ the *weak CR property*.

1. $\mathrm{cr}_R(a) \Leftrightarrow \forall b_1, b_2 \in A.[aR^*b_1 \ \& \ aR^*b_2 \ \Rightarrow \ \exists c.b_1R^*c \ \& \ b_2R^*c]$.
2. $\mathrm{CR}(R) \Leftrightarrow \forall a \in A.\mathrm{cr}_R(a)$.
3. $\mathrm{WCR}(R) \Leftrightarrow \forall a, b_1, b_2 \in A.[aRb_1 \ \& \ aRb_2 \ \Rightarrow \ \exists c.b_1R^*c \ \& \ b_2R^*c]$.

Newman's lemma states that for well-founded relations weak confluence implies strong confluence. The notion of well-foundedness is formulated as the possibility to prove statements by transfinite induction. Let $P \in \mathcal{P}(A)$.

4. $\mathrm{IND}_R(P) \Leftrightarrow \forall a{\in}A.(\forall y{\in}A.aRy \Rightarrow P(y)) \Rightarrow P(a)$.

5. $\mathrm{WF}(R) \Leftrightarrow \forall P{\in}\mathcal{P}(A).[\mathrm{IND}_R(P) \Rightarrow \forall a{\in}A.P(a)]$.

3.1. Lemma. (i) $\forall x, y{:}A.[xR^*y \Rightarrow (x = y \vee xR^+y)]$.
(ii) $\forall x, y{:}A.[xR^+y \Rightarrow \exists z{:}A.(xRz \,\&\, zR^*y)]$.

3.2. Lemma (Main Lemma.). $\mathrm{WCR}(R) \Rightarrow \mathrm{IND}_R(cr_R)$.

Proof. Assume WCR(R). Remember $IND_R(cr_R) \Leftrightarrow$

$$(\forall a{:}A.(\forall y{:}A.a\ R\ y{\rightarrow}cr(R,y)){\rightarrow}(cr_R(a)).$$

Let $a{:}A$ and assume

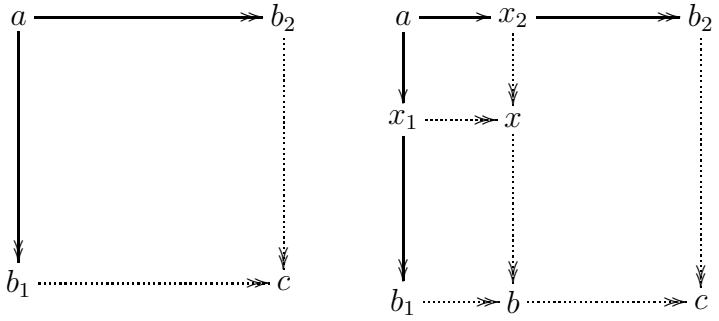
$$\forall y{:}A.a\ R\ y{\rightarrow}cr_R(y), \qquad (IH)$$

in order to show $\mathrm{cr}_R(a)$, i.e.

$$\forall b_1, b_2{:}A.a\ R^*\ b_1 \ \&\ a\ R^*\ b_2{\rightarrow}(\exists c{:}A.b_1\ R^*\ c \ \&\ b_2\ R^*\ c).$$

So let $b_1, b_2{:}A$ with $a\ R^*\ b_i$, in order to show $\exists c.b_i\ R^*\ c$.

If $a = b_1$ or $a = b_2$, then the result is trivial (take $c = b_2$ or $c = b_1$ respectively). So by lemma 3.1(ii) we may assume $a\ R^+\ b_i$, which by lemma 3.1(i) means $a\ R\ x_i\ R^*\ b_i$, for some x_1, x_2.



By $\mathrm{WCR}(R)$ there is an x such that $x_i\ R^*\ x$.
By (IH) one has $cr_R(x_1)$. So $x\ R^*\ b \ \&\ b_1\ R^*\ b$, for some b.
Again $cr_R(x_2)$. As $x_2\ R^*\ x\ R^*\ b$ one has $b\ R^*\ c \ \&\ b_2\ R^*\ c$, for some c.
Then $b_1\ R^*\ b\ R^*\ c$ and we are done. ■

3.3. Proposition (Newman's Lemma). *WCR(R) & WF(R) ⇒ CR(R).*

Proof. By $\mathrm{WCR}(R)$ and the main lemma we have $\mathrm{IND}_R(\mathrm{cr}_R)$. Hence by $\mathrm{WF}(R)$ it follows that for $P(a) = \mathrm{cr}_R(a)$, one has $\forall a{\in}A.\mathrm{cr}_R(a)$. This is $CR(R)$. ■

3.2. Proof sketch for Newman's lemma

Now we will start a proof development for Newman's lemma.

```
Variable A:Set.
Definition Bin:=[B:Set](B->B->Prop).

Inductive TC [R:(Bin A)]: (Bin A) :=
TCb:  (x,y:A)(R x y)->(TC R x y)|
TCf:  (x,y,z:A)((R x z)->(TC R z y)->(TC R x y)).

Inductive TRC [R:(Bin A)]: (Bin A) :=
TRCb:  (x:A)(TRC R x x)|
TRCf:  (x,y,z:A)((R x z) -> (TRC R z y)->(TRC R x y)).

Definition Trans [R:(Bin A)]: Prop:=
(x,y,z:A)((R x y)->(R y z)->(R x z)).

Definition IND [R: (Bin A);P:(A->Prop)]:  Prop :=
((a:A)((y:A)(a R y)->(P y))->(P a)).

Definition cr [R:(Bin A);a:A]:=
(b1,b2:A)(TRC R a b1)&(TRC R a b2)->(EX c:A|(TRC R b1 c)&(TRC R b2 c)).

Definition CR [R:(Bin A)]:=(a:A)(cr R a).

Definition WCR [R:(Bin A)]:=
(a,b1,b2:A)(a R b1)->(a R b2)->(EX c:A|(TRC R b1 c)&(TRC R b2 c)).

Definition WF [R:(Bin A)]:Prop:= (P:A->Prop)(IND R P)->(a:A)(P a).

Variable R:(Bin A).
```

First we introduce some userfriendly notation.

NOTATION. `For a,b:A we write`

(i) `a R  b := (R a b).`
(ii) `a R+ b := (TC R a b).`
(iii) `a R* b := (TRC R a b).`

The following lemmas can be proved formally.

```
Lemma p1:  (x,y:A)((x R y) -> (x R+ y)).
Lemma p2:  (x,y:A)((x R+ y) -> (x R* y)).
Lemma p3:  (Trans(R+)).
Lemma p4:  (Trans(R*)).
Lemma p5:  (x,y,z:A)(x R y)->(y R* z)->(x R* z).
Lemma p6:  (x,y:A)((x R* y)-> (eq A x y) ∨ (x R+ y)).
Lemma p7:  (x,y:A)((x R+ y)->(EX z:A | (x R z) & (z R* y))).
```

Now we will give a mathematical proof script of the main lemma, for which we claim that it should be acceptable by a mathematician-friendly proof-assistant. At `<www.cs.kun.nl/~henk/mathmode.ps>` one may view an interactive version of this script that conveys better the power of the proofmode. If one has active dvi, then one should look at `<mathmode.dvi>` at the same address.

```
Lemma (WCR R)->(IND R (cr R)).

Proof.  Assume WCR(R). Remember IND. Let a:A. Assume
        (y:A)((aRy)->(cr R y)).                                  (IH)

Remember cr.  Let a,b1,b2:A. Assume a R* bi, i=1,2,
in order to show (EX c:A | (b1 R* c) & (b2 R* c)).
We have, by lemma p6,
     [a=b1 ∨ a R+ b1],
     [a=b2 ∨ a R+ b2].
Case a=b1, take c=b2.  Trivial.  Hence wlog (a R+ b1).
Case a=b2, take c=b1.  Trivial.  Hence wlog (a R+ b2).
Therefore, by lemma p7, there exists x1, x2 such that
     a R xi R* bi, i=1,2.
Hence, by (WCR R), there exists x such that xi R* x, i=1,2.
We have (cr R x1), by IH.
Hence there exists b such that b1 R* b & x R* b.
Moreover (cr R x2), by IH. Hence, by x2 R* b, there exists c
such that b R* c & b2 R* c.  Since b1 R* c, by (Trans R*),
we have (bi R* c), i=1,2.  Thus c works.  QED

Newman's Lemma.  WCR(R)&WF(R) -> CR(R).
```

```
Proof.  Assume WCR(R) and WF(R). Then (IND R( cr R)),
by WCR(R) and main.  Remember IND. We have
(P:(A->Prop))((a:A)((y:A)(a R y)-> (P y))->(P a)).          (+)
Apply (+) to P=(cr R). Then CR(R). QED
```

4. Towards a Mathematical Proof Language

We now will sketch rather loosly a language that may be called MPL: *Mathematical Proof Language*. The language will need many extensions, but this kernal may be already useful.

4.1. Definition. The phrases used in MPL for the proposed proof-assistant with interactive mathematical mode belong to the following set.

```
Assume B
Towards A
Let x:D
Remember t
Pick [in L] x
Take x=t [in B]
Apply B to t
Case B
As to
There exists x
QED

Then B [, by C]
Suffices
Wlog B, [since B ∨ C]
May assume B
```

Here `A,B, C` are propositions in context `Gamma`, `D` is a type, `x` is a variable and `t` is a term of the right type. "`Wlog`" stands for "Without loss of generality".

4.2. Definition (Synonyms).

```
Suffices = In order to show = We must show = Towards;
Let = Given;
Then = We have = It follows that = Hence = Moreover = Again;
```

```
and = with;
by = since.
```

Before giving a grammar for tactic statements we will give their semantics. They have a precise effect on the so called proof-state. In the following definition we show what the effect is of a statement on the proof-state. In some of the cases the tactic has a side-effect on the proof-script, as we already saw in the case study of Newman's lemma.

4.3. DEFINITION. (i) A *proof-state* (within a context `Gamma`) is a set of statements `Delta` and a statement `A`, such that all members of `Delta` are well-formed in `Gamma` and `A` is well-formed in `Gamma, Delta`. If the proof-state is `(Delta;A)`, then the goal is to show `Delta` $\vdash$ `A`.

(ii) The *initial* proof-state of a statement `A` to be proved is of course `(`$\emptyset$`;A)`.

(iii) A tactic is map from proof-states to a list of proof-states, usually having a formula or an element as extra argument.

4.4. DEFINITION. Various tactics are defined as follows.

```
Assume C (Delta,C->B)          =  (Delta,C;B), and ‘‘Towards B’’
                                  may be left in the script.
Let a:D (Delta,(x:D.P))        =  (Delta,a:A;P[x:=a]).
Remember name (Delta;A)        =  (Delta;A’), where A’ results
                                  from A by unfolding the defined
                                  concept ‘name’.  This can be
                                  applied to an occurrence of
                                  ‘name’, by clicking on it.  Other
                                  occurrences remain closed but
                                  become transparant (as if opened).
Pick [in L] x (Delta,L;A)      =  (Delta,x:D,B(x);A), where L is
                                  a formula reference of (EX x:D.B).
Take x=name (Delta;EX x:D.A)   =  (Delta;A[x:=name]),
                                  if Delta |- name:D.
Apply B to name (Delta;A)      =  (Delta,P[y:=name];A), where B of
                                  the form ((y:D).P) is in Delta.
Case B (Delta;A)               =  (Delta,B;A),(Delta,C;A),
                                  if B ∨ C in Delta; the second
                                  proof-state represents
                                  the next subgoal.
```

```
As to Bi (Delta;B0 & B1)     =  (Delta;Bi),(Delta;B(1-i)), the
                                second proof-state represents
                                the next subgoal;
As to B (Delta;B)            =  (Delta;B);
There exists x such that A
     (Delta, (EX x:D.A); B)  =  (Delta,x:D,A;B).
```

In all cases nothing happens if the side conditions are not satisfied. One should be able to refer to a statement `C` in two ways: either by naming `C` directly or by referring to a label for `C`, like "`IH`" in the proof of the main lemma above. We say that L is a formula reference of formula B if L is B or if L is a label for B. Labels are sometimes handy, but they should also be suppressed in order to keep the proof-state clean. If the argument of a tactic occurs at several places the system should complain. Then reference should be made to a unique label. It is assumed that proof-states `(Delta,A)` are in normal form, that is, if `B&C` is in `Delta`, then it is replaced by the pair `B,C`. If the final `QED` is accepted, then all the statements in the proof that did not have an effect on the proof-state will be suppressed in the final lay-out of the proof (or may be kept in color orange as an option in order to learn where one did superfluous steps).

The following tactics require some automated deduction. If the proof-assistant cannot prove the claimed result, an extra proof-state will be generated so that this result will be treated as the next subgoal.

4.5. Definition.

```
Then B (Delta;A)             =  (Delta,B;A), if Delta |- B;
Then B, by C (Delta;A)       =  (Delta,B;A) if Delta |- C and
                                Delta,C |- B;
Suffices B (Delta;A)         =  (Delta; B), if Delta |- B->A;
Wlog C (Delta;A)             =  (Delta,C;A), if Delta |- C;
Since B∨C wlog C (Delta;A)   =  (Delta,C;A), if B∨C in Delta
                                and Delta|-B->A.
May assume B (Delta,A)       =  (Delta,B;A) if the assistant
                                Delta|-~B->A and Delta|- B∨~B.
```

The tactic language MPL is defined by the following grammar.

4.6. Definition. Grammar of MPL.

```
formref  :=  label | form
  form+  :=  formref | form+ and formref
```

```
 tactic  :=  Assume form+ | Towards form | Remember name |
             Let var:set | Pick [in formref] var | Case form |
             Take var = term [in formref ] |
             Apply formref to term | Then form[, by form+] |
             Suffices formref | Wlog form[, since form ∨ form]
tactic+  :=  tactic.  | tactic, tactic+ | tactic.  tactic+
```

Here label is the proof-variable, used as a name for a statement (like IH in the proof of the main lemma), form is a Gamma, Delta inhabitant of Prop, name is any defined notion during the proof development, and var is an variable.

An extension of MPL capable of dealing with computations will be useful.

```
We have A(t).  Then A(s), since t=s.
```

Another one:

```
Then t=s, by computation.
```

It would be nice to have this in an ambiguous way: computation is meant to be pure conversion or an application of reflection. This corresponds to the actual mathematical usage:

5!=120, by computation;
In a commutative ring, $(x+y)^2 = x^2 + 2xy + y^2$, by computation.

In typetheory the first equality would be an application of the conversion rule, but for the second one reflection, see e.g. Constable [1995], is needed.

Procedural statements in the implementation of MPL

As we have seen in section 2 it is handy to have statements that modify the proofstate, but are not recorded as such. For example if the proof state is `(Delta;(x:D)(A(x)->B(x))`, then `Intros` is a fast way to generate

```
Let x:D. Assume A(x), in order to prove B(x).
```

in the proof. Another example is `Clear L` in order to remove the formula indicated by `L` in the assumptions of the current subgoal. Also renaming variables is useful, as some statements may come from libraries and have a "wrong" choice of bound variables.

5. Conclusion

We claim that an interactive mathematical mode is necessary for proof-assistants, if they are to be used in a user friendly way. It is hoped that in this way also mathematicians (next to the computer scientists) will become users of these systems. As has been argued elsewhere the use of mathematical assistants will be able to modify the way mathematics is done and achieve a higher standard of precision. The present proof sketch may serve as part of a benchmark for such an improved mathematical assistant.

In Nederpelt [2002] a language WTT is developed with similar aims as our language MPL. So far the development of WTT has been focussed on the theory development: the formulation of the concepts and of the statements of the lemmas. Hence it is a welcome (and much more developed) complement to MPL. A combination of WTT and MPL is what is needed for a proof-assistant with interactive mathematical mode.

References

de Bruijn, N.G. [1994]. The mathematical vernacular, a language for mathematics and typed sets, in *Nederpelt et al.* [1994], pp. 865–936.

Constable, R.L. [1995]. Using reflection to explain and enhance type theory, *in:* H. Schwichtenberg (ed.), *Proof and Computation*, Computer and System Sciences 139, Springer, pp. 109–144.

Constable, Robert L., Stuart F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, Douglas J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, James T. Sasaki and Scott F. Smith [1986]. *Implementing Mathematics with the Nuprl Development System*, Prentice-Hall, NJ.

Gordon, M.J.C. and T.F. Melham (eds.) [1993]. *Introduction to HOL*, Cambridge University Press, Cambridge.

Harrison, John [2000]. *The HOL Light manual (1.1).* URL:
`<www.cl.cam.ac.uk/users/jrh/hol-light/manual-1.1.ps.gz>`.

Muzalewski, M. [1993]. *An Outline of PC Mizar*, Fondation Philippe le Hodey, Brussels. URL:
`<www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz>`.

Nederpelt, R. [2002]. Weak Type Theory: a formal language for mathematics, *Technical Report 02-05*, Dept. of Mathematics and Computer Science, Eindhoven University of Technology, Box 513, 5600 MB Eindhoven, The Netherlands. URL:
<vubisweb.tue.nl/N/scripts/mgwms32.dll?TS=SA>.

Nederpelt, R. P., J. H. Geuvers and R. de Vrijer (eds.) [1994]. *Selected papers on Automath*, North-Holland Publishing Co., Amsterdam.

Nipkow, T., L.C. Paulson and M. Wenzel [2002]. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, LNCS 2283, Springer.

Paulson, L.C. [1994]. *Isabelle: a generic theorem prover*, LNCS 828, Springer-Verlag, New York.

Paulson, L.C. [2002]. *The Isabelle Reference Manual*. URL:
<isabelle.in.tum.de/doc/ref.pdf>.

The Coq Development Team [2002]. *The Coq Proof Assistant Reference Manual*. URL:
<ftp.inria.fr/INRIA/coq/current/doc/Reference-Manual-all.ps.gz>.

Wenzel, M. [2002a]. *Isabelle/Isar — a versatile environment for human-readable formal proof documents*, Dissertation, Institut für Informatik, Technische Universität München. URL:
<tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/wenzel.html>.

Wenzel, M. [2002b]. *The Isabelle/Isar Reference Manual*, TU München. URL:
<isabelle.in.tum.de/doc/isar-ref.pdf>.

Wenzel, Markus [1999]. Isar — a generic interpretative approach to readable formal proof documents, *in:* Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin and L. Thery (eds.), *Theorem Proving in Higher Order Logics: TPHOLs '99*, LNCS 1690.

Wiedijk, F. [1999]. Mizar: An Impression, URL:
<www.cs.kun.nl/~freek/mizar/mizarintro.ps.gz>.