The formal proof sketch challenge

Freek Wiedijk

University of Nijmegen

Abstract. A *formal proof sketch* is a way to present a formal proof in a style that is close to an informal proof, but which also is a skeleton of the full formal proof (this makes it easy to relate the presentation to the detailed formalization.) Recently to us every informal proof has started to feel like a *challenge*, to write down the corresponding formal proof sketch. We take on this challenge for the informal proof of *Newman's lemma* from Henk Barendregt's book about λ-calculus. The specific difficulty of that proof is its main part, which just is a pair of diagrams without any explanation.

1 Introduction

1.1 Problem

Often when encountering a nice piece of mathematics, we feel a strong desire to formalize it. This feeling probably is similar to the desire of an artist to create a beautiful painting when he sees a beautiful woman.

Recently we introduced the notion of a formal proof sketch [7]. For us the urge to find the formal proof sketch corresponding to a specific informal proof turns out to be even stronger than the desire to create a formalization in some less direct way. In [7] we presented formal proof sketches for two small proofs only (a proof of the irrationality of $\sqrt{2}$ from Hardy & Wright [5] and a proof of a small lemma from linear algebra), but now every informal proof that we encounter seems to demand that it wants a formal proof sketch of its own. We call this urge that makes us want to find out what formal proof sketches shaped after informal proofs look like, the *formal proof sketch challenge.* The challenge then, is to get as close as possible to the original.

Henk Barendregt told us that he thought that a good example for experimenting with formal proof sketches would be the proof of Newman's lemma (a simple lemma about confluence from the theory of abstract rewriting). In our turn we thought that it would be interesting to investigate the specific proof of Newman's lemma that Henk gave in his classic textbook about λ-calculus [1]. The result of this investigation is this note.

A further reason to be interested in this particular proof is that Henk told us that he thought he remembered that at some point he was not able to explain it to N.G. de Bruijn. If this indeed has happened, then apparently the proof from [1] is not completely trivial. Maybe the formal proof sketch from this paper can help making that specific version of the proof more understandable to people who have a formalist bend.

1.2 Approach

We made a formal proof sketch of Henk's version of Newman's lemma following the same approach that we used in [7]. That is, we made a formal proof sketch using the exact syntax of the current version of the Mizar system, and we even tried to be 'compatible' with the big Mizar library that accompanies the Mizar system (the 'Mizar mathematical library' or MML).

1.3 Related Work

The proof of Newman's lemma is very simple. This makes it one of the basic examples for all of the major proof checking systems, like for instance ACL2, Coq and Isabelle. In the Mizar mathematical library Newman's lemma is proved in article `REWRITE1` by Grzegorz Bancerek.

Diagrammatic reasoning is an active field of research. An interesting system that implements proofs both involving logical steps and diagrams, is the Hyperproof system [2] by Jon Barwise and John Etchemendy. In [3] they explain the philosophy behind this system.

1.4 Contribution

We give a formal proof sketch for an informal proof that is not primarily textual. We also present the full formalization that one gets by 'completing' the formal proof sketch.

Despite the non-textual nature of the original, we show that the formal proof sketch manages to closely follow the informal proof. This shows that Mizar formal proof sketches are a versatile way to present formal mathematics in an informally acceptable way.

When discussing the relation between the diagram and the corresponding part of the formal proof sketch, we show that a diagram is a dynamic entity that actually consists of a series of growing diagrams. We explain a mechanism that a proof assistant can use to automatically generate diagrams from a proof.

We discuss the relation between the presentation of our formal proof sketch and the treatment of rewriting that is in the Mizar mathematical library, both conceptually and notationally. We compare Henk's version of the proof to the proof of Newman's lemma that is in the Mizar mathematical library.

1.5 Outline

In Section 2 we present both the informal proof from [1] and the corresponding formal proof sketch. In Section 3 we discuss the relation between the diagram from the original and the steps in the formal proof sketch. In Section 4 we present the full formalization of the proof, together with the statements of the lemmas that it uses. In Section 5 we discuss the relation between the notation that is used in the Mizar mathematical library and our notation. In Section 6 we present the proof of Newman's lemma from the Mizar mathematical library by way of another – this time *extracted* – formal proof sketch.

2 A formal proof sketch of Newman's lemma

Here is a copy of part of page 58 from [1]:

informal proof

3.1.25. Proposition. *For notions of reduction one has*

$$\mathrm{SN} \wedge \mathrm{WCR} \Rightarrow \mathrm{CR}$$

Proof. By SN each term R-reduces to an R-nf. It suffices to show that this R-nf is unique. Call M *ambiguous* if M R-reduces to two distinct R-nf's. For such M one has $M \to_R M'$ with M' ambiguous (use WCR, see figure 3.3). Hence by SN ambiguous terms do not exist.

$$\begin{tikzcd}
 & & M \arrow[dl] \arrow[dr] & & \\
 & M' \arrow[ddl, two heads] \arrow[dr, two heads] & & {} \arrow[dl, two heads] \arrow[ddr, two heads] & \\
 & & {} \arrow[d, two heads] & & \\
M_1 & & M_3 & & M_2
\end{tikzcd}$$

or

$$\begin{tikzcd}
 & M \arrow[d] & \\
 & M' \arrow[d, two heads] & \\
 & {} \arrow[dl, two heads] \arrow[dr, two heads] & \\
M_1 & & M_2
\end{tikzcd}$$

fig. 3.3.

$\square$

This is a proof of the statement known as *Newman's lemma* [6]. It relates some properties of abstract 'notions of reduction' (a binary relation looked at as the abstraction of a rewrite system). In this, the abbreviation 'SN' stands for 'strongly normalizing' (which means that there is no infinite reduction sequence), 'CR' stands for the 'Church-Rosser' property (which means that reducts of a common original always can be reduced in a converging way, i.e., to a common reduct), and 'WCR' stands for 'weakly Church-Rosser' (which means that this Church-Rosser property only has to hold when the diverging reductions are only one step long).

And here is the corresponding 'formal proof sketch' (a notion which we introduced in [7]):

formal proof sketch

```
    reserve R for nf_inhabited notion_of_reduction;
    reserve M,M',M'',M''' for Term of R;
    reserve M1,M2,M3 for Nf of R;

   theorem 3_1_25:
    R is SN & R is WCR implies R is CR
   proof
    assume that R is SN and R is WCR;
    for M ex M1 st M reduces_to M1;
    (for M,M1,M2 st M reduces_to M1 & M reduces_to M2 holds M1 = M2)
     implies R is CR;
    defpred ambiguous[Term of R] means
     ex M1,M2 st $1 reduces_to M1 & $1 reduces_to M2 & M1 <> M2;
    now
     now
      let M such that ambiguous[M];
      thus ex M' st M ---> M' & ambiguous[M']
      proof
       consider M1,M2 such that M -->> M1 & M -->> M2 & M1 <> M2;
       per cases;
       suppose not ex M' st M ---> M' & M' -->> M1 & M' -->> M2;
        consider M' such that M ---> M' & M' -->> M1;
        consider M'' such that M ---> M'' & M'' -->> M2;
        consider M''' such that M' -->> M''' & M'' -->> M''';
        consider M3 such that M''' -->> M3;
        take M';
        thus thesis;
       suppose ex M' st M ---> M' & M' -->> M1 & M' -->> M2;
        consider M' such that M ---> M' & M' -->> M1 & M' -->> M2;
        take M';
        thus thesis;
      end;                                                       :: fig. 3.3
     end;
     thus not ex M st ambiguous[M];
    end;
    thus thesis;
   end;
```

Note that we have used the *variable convention* here that the variables called M_1, M_2, M_3, ... are referring to normal forms. Mizar has the `reserve` statement to introduce such variable conventions. Alternatively we could have put '`being Nf of R`' at each place where such a variable was introduced. That would have made the formal proof sketch more explicit (and therefore maybe a bit easier to follow), but also more cluttered.

The 'notion of reduction' that this formal proof sketch talks about has the attribute `nf_inhabited`. Mizar only allows non-empty types and we wanted to use a type `Nf of R` for the normal forms of `R`. Therefore we only prove the theorem for notions of reduction that have one or more normal forms. (In a sense that does not restrict the theorem, because all `SN` notions of reduction are `nf_inhabited`.) One could get rid of the `nf_inhabited` by not using the `Nf of R` type, but replacing all quantifiers like:

```
for M1 being Nf of R holds ...
```

by corresponding quantifiers like:

```
for M1 being Term of R st M1 is_nf holds ...
```

The part of the proof in the box corresponds to the diagram from the original. One gets a second, smaller, formal proof sketch of the same original by replacing the text in the box by a single semi-colon. To show that this shortened formal proof sketch corresponds very closely to the original text, we repeat the two versions here, interleaved with each other:

Proof.

```
proof
 assume that A1: R is SN and R is WCR;
```

By SN each term R-reduces to an R-nf.

```
 for M ex M1 st M reduces_to M1 by A1;
```

It suffices to show that this R-nf is unique.

```
 (for M,M1,M2 st M reduces_to M1 & M reduces_to M2 holds
   M1 = M2) implies R is CR;
```

Call M *ambiguous* if M R-reduces to two distinct R-nf's.

```
 defpred ambiguous[Term of R] means
  ex M1,M2 st $1 reduces_to M1 & $1 reduces_to M2 & M1 <> M2;
```

For such M one has $M \to_R M'$ with M' ambiguous.

```
 now
  now
   let M such that ambiguous[M];
   thus ex M' st M ---> M' & ambiguous[M'];
  end;
```

Hence by SN ambiguous terms do not exist.

```
  thus not ex M st ambiguous[M] by A1;
 end;
```

□

```
   thus thesis;
  end;
```

It should be noted that we did not create this formal proof sketch by first doing a full formalization, and then cutting out the sketch afterward. Instead we wrote the formal proof sketch directly, and only after-wards completed it to a full formalization (see Section 4). It is one of the strengths of the Mizar language that this is possible, and in fact even easy (the difficult part of doing a Mizar formalization is finding the proper references to the Mizar mathematical library.)

3 Proofs and diagrams

The part of the formal proof sketch that corresponds to the diagram from the original, in fact corresponds to a *sequence* of growing diagrams. This means that the diagram in 'fig. 3.3' is not one diagram but a whole series. A paper book can not show this series (at least not without using more paper than is worth it), but an interactive proof in a computer can. To be able to see these 'living diagrams' might be a good reason for wanting to formalize mathematics.

Between each subsequent pair of steps in the proof there is a 'natural' diagram that shows the situation between the steps:

$$M$$

```
consider M1,M2 such that M -->> M1 & M -->> M2 & M1 <> M2;
```

$$\begin{tikzcd} & M \arrow[dl] \arrow[dr] & \\ M_1 & & M_2 \end{tikzcd}$$

```
consider M' such that M ---> M' & M' -->> M1;
```

$$\begin{tikzcd} & & M \arrow[dl] \arrow[ddrr] & & \\ & M' \arrow[dl, two heads] & & & \\ M_1 & & & & M_2 \end{tikzcd}$$

```
consider M'' such that M ---> M'' & M'' -->> M2;
```

$$\begin{tikzcd} & & M \arrow[dl] \arrow[dr] & & \\ & M' \arrow[dl, two heads] & & M'' \arrow[dr, two heads] & \\ M_1 & & & & M_2 \end{tikzcd}$$

```
consider M''' such that M' -->> M''' & M'' -->> M''';
```

$$\begin{tikzcd}
& & M \arrow[dl] \arrow[dr] & & \\
& M' \arrow[dr, two heads] \arrow[ddl, two heads] & & M'' \arrow[dl, two heads] \arrow[ddr, two heads] & \\
& & M''' & & \\
M_1 & & & & M_2
\end{tikzcd}$$

```
consider M3 such that M''' -->> M3;
```

$$\begin{tikzcd}
& & M \arrow[dl] \arrow[dr] & & \\
& M' \arrow[dr, two heads] \arrow[ddl, two heads] & & M'' \arrow[dl, two heads] \arrow[ddr, two heads] & \\
& & M''' \arrow[d, two heads] & & \\
M_1 & & M_3 & & M_2
\end{tikzcd}$$

These diagrams can be generated automatically by a proof assistant in the following way:

At every point in the proof there is a proof context (or proof goal) consisting of the statement that needs to be proved (what Mizar calls the 'thesis'), the *variables* that have been introduced by `let` and `consider` statements, and the *statements* in the proof that can be referred to.

Now one can select from those variables the ones that have a certain type (in the case of the example: `Term of` ...) and the from those statements the statement that involve those variables and that have a certain shape (in the case of the example: ... `--->` ... and ... `-->>` ...). Both sets are steadily growing through a proof. A *diagram* is a graphical representation of both sets. In the example it consists of a graph with the variables at the vertices and the two kinds of predicates represented by two kind of edges. Of course this graph needs to be laid out in the plane, but there exist algorithms that can do that automatically (a nice example of an implementation of such an algorithm [4] is the `dot` program from AT&T's `graphviz` system).

Note that in a proof by contradiction the set of relevant statements might become inconsistent. It needs investigation how to pleasantly represent such a situation in a diagram. Note that the diagrams might be interactive: instead of having the proof assistant determining the layout of the diagram, the user might interactively drag elements of the diagram around. Note that the diagrams can be graphically consistent along the series (so later diagrams are obtained by adding elements to earlier diagrams, without moving the elements that are already present), but they do not need to be. Finally note that for big proofs it might be advantageous to leave out part of the proof context from the diagram, either under control of the algorithms generating the diagram from the proof context or under control of the user.

In this example the diagrams represent reductions in an abstract reduction system but the basic principles are the same for other kinds of diagrams like geometrical diagrams, diagrams from graph theory or commuting diagrams from category theory.

4 The full formalization

Here is the full formalization that one gets by 'completing' the formal proof sketch from Section 2. The parts that already were present in the formal proof sketch have been underlined:

full formalization

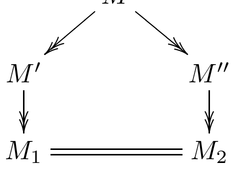
```
 reserve R for nf_inhabited notion_of_reduction;
 reserve M,M',M'',M''' for Term of R;
 reserve M1,M2,M3 for Nf of R;

theorem 3_1_25:
 R is SN & R is WCR implies R is CR
proof
 assume that
A1: R is SN and
A2: R is WCR;
A3: R is WN by A1,Th9;
 then for M ex M1 st M reduces_to M1 by Def10;
A4: (for M,M1,M2 st M reduces_to M1 & M reduces_to M2 holds M1 = M2)
  implies R is CR
 proof
  assume
A5: for M,M1,M2 st M reduces_to M1 & M reduces_to M2 holds M1 = M2;
  let M,M',M'';
  assume
A6: M -->> M' & M -->> M'';
  consider M1 such that
A7: M' -->> M1 by A3,Def10;
  consider M2 such that
A8: M'' -->> M2 by A3,Def10;
  M -->> M1 & M -->> M2 by A6,A7,A8,Th6;
  then M' -->> M1 & M'' -->> M1 by A5,A7,A8;
  hence thesis;
 end;
 defpred ambiguous[Term of R] means
  ex M1,M2 st $1 reduces_to M1 & $1 reduces_to M2 & M1 <> M2;
A9: now
A10: now
   let M such that
A11: ambiguous[M];
```

```
      thus ex M' st M ---> M' & ambiguous[M']
      proof
       consider M1,M2 such that
A12:    M -->> M1 & M -->> M2 & M1 <> M2 by A11;
       per cases;
       suppose
A13:    not ex M' st M ---> M' & M' -->> M1 & M' -->> M2;
        M1 is_nf & M2 is_nf by Def9;
        then
A14:    M <> M1 & M <> M2 by A12,Th8;
        then consider M' such that
A15:    M ---> M' & M' -->> M1 by A12,Th7;
        consider M'' such that
A16:    M ---> M'' & M'' -->> M2 by A12,A14,Th7;
        consider M''' such that
A17:    M' -->> M''' & M'' -->> M''' by A2,A15,A16,Def11;
        consider M3 such that
A18:    M''' -->> M3 by A3,Def10;
        take M';
        M' -->> M3 & M'' -->> M3 by A17,A18,Th6;
        then M' -->> M1 & M' -->> M3 & M1 <> M3 by A13,A15,A16;
        hence thesis by A15;
       suppose ex M' st M ---> M' & M' -->> M1 & M' -->> M2;
        then consider M' such that
A19:    M ---> M' & M' -->> M1 & M' -->> M2;
        take M';
        thus thesis by A12,A19;
      end;
     end;
     thus not ex M st ambiguous[M] from SN_induction1(A1,A10);
    end;
    thus thesis by A4,A9;
   end;
```

Note that the statement from the original:

It suffices to show that this R-nf is unique.

needs a multi-line sub-proof here. That proof corresponds to the diagram:


$$\begin{tikzcd}
 & M \arrow[dl, two heads] \arrow[dr, two heads] & \\
M' \arrow[d, two heads] & & M'' \arrow[d, two heads] \\
M_1 \arrow[rr, equal] & & M_2
\end{tikzcd}$$

The formalization uses the following basic lemmas:

```
theorem Th6: M -->> M’ & M’ -->> M’’ implies M -->> M’’;

theorem Th7: M -->> M’’ implies
 M = M’’ or ex M’ st M ---> M’ & M’ -->> M’’;

theorem Th8: M is_nf & M -->> M’ implies M = M’;

theorem Th9: R is SN implies R is WN;

scheme SN_induction1 {R()->notion_of_reduction, X[set]}:
 for M being Term of R() holds not X[M]
provided
 R() is SN and
 for M being Term of R() st X[M]
  ex M’ being Term of R() st M ---> M’ & X[M’];
```

We considered these statements general enough that their proofs should not be part of the proof of Newman’s lemma.

The last statement `SN_induction1`, is equivalent to Noetherian induction:

```
scheme SN_induction {R()->notion_of_reduction, X[set]}:
 for M being Term of R() holds X[M]
provided
 R() is SN and
  for M being Term of R() st
   for M’ being Term of R() st M ---> M’ holds X[M’]
  holds X[M];
```

Although this looks different from `SN_induction1`, it is exactly the same when one replaces `X[M]` by `not X[M]`. Both are higher order statements. Mizar is a first order language, but it knows this kind of higher order statements as *schemes*. Mizar has the language of set theory, so it can state the same thing in a first order way as well:

```
attr R is SN means
 for X being Subset of field R st
  (for M st for M’ st M ---> M’ holds M’ in X holds M in X) holds
   for M holds M in X;
```

However, because Mizar has no good λ-abstraction, the schematic form is easier to apply and therefore it is the one that we used in the formalization. Note that this lack of proper λ-abstraction is not a problem for making formal proof sketches look like informal mathematics, because in informal mathematics λ-notation is almost never used.

5 Notation and the Mizar mathematical library

The big Mizar library called MML – Mizar mathematical library – already contains a treatment of rewriting in article `REWRITE1`. However the notation used there is not exactly the same as the one used in the proof from Henk's book. Mizar has the possibility to define *synonyms*, so we were able to make the notation closer to the one in the original than if we had just used the notation that was already present in the MML. In this table we show in the first column the notation used in Henk's proof, in the second column the notation for this from the MML, and in the third column the notation that we used:

Henk's book	*MML*	*formal proof sketch*
notion of reduction	`Relation`	`notion_of_reduction`
term	`set`	`Term of R`
R is SN	`R is strongly-normalizing`	`R is SN`
R is WCR	`R is locally-confluent`	`R is WCR`
R is CR	`R is confluent`	`R is CR`
$M \to_R M'$	`[M,M'] in R`	`M ---> M'`
M R-reduces to M'	`R reduces M,M'`	`M reduces_to M'`
M is a R-nf	`M is_a_normal_form_wrt R`	`M is_nf`
M is ambiguous		`ambiguous[M]`

There were two design choices for modeling `notion_of_reduction` in our formalization:

– In the MML there are two types that can be used to model an abstract reduction system. In article `RELAT_1` the type `Relation` is defined to be an arbitrary set of pairs. This is the concept that is used in article `REWRITE1`. In article `ORDERS_1` the type `RelStr` ('relational structure') is defined by:

```
definition
 struct(1-sorted) RelStr (#
  carrier -> set,
  InternalRel -> Relation of the carrier
 #);
end;
```

  Here one separates the carrier of the reduction structure from the reduction relation itself. This is more accurate when one wants to model notions of reduction, but we wanted to reuse as much from `REWRITE1` as possible, so we chose to define `notion_of_reduction` as:

```
definition
 mode notion_of_reduction is non empty Relation
end;
```

– In Mizar a predicate takes two comma-separated lists of arguments (possibly single or maybe even empty) on the left and the right of the relation symbol. Using that format it is hard to imitate the $M \to_R M'$ notation (it would become `M ---> R,M'`, which is ugly). We chose to omit the R from this notation, and have the current reduction relation *inferred* from the type of the terms. In order to make this work, the type needs to be a dependent type `Term of R`:

```
definition let R be notion_of_reduction;
 mode Term of R is Element of field R;
end;
```

Using this type, one defines the reduction arrow by:

```
definition let R be notion_of_reduction;
 let M,M' be Term of R;
 pred M ---> M' means [M,M'] in R;
end;
```

The `R` can be reconstructed from the types of the `M` and `M'`, so it does not need to be part of the notation.
The choice to use dependent types (to be able to omit the reduction relation from the notation) causes the attributes like `SN` and so on to become 'non clusterable'. This means that Mizar is not able to infer attributes by itself using so-called 'cluster rules'. For instance, we needed to state the fact that `R is WN` explicitly in our proof (statement `A3`). If the attributes had been clusterable, we also could have phrased the statement of the theorem as a 'cluster':

```
cluster SN WCR -> CR notion_of_reduction
```

which looks more like the statement of the original informal proof:

$$\mathrm{SN} \wedge \mathrm{WCR} \Rightarrow \mathrm{CR}$$

than what is now in the formal proof sketch.

Our decision to be compatible with the MML blocked some notational possibilities. The symbol `term` is already used in the MML for a function, so it can not be used for a type as well, and so we can not write `term of R`. Instead we used `Term of R`. (Generally types are written with capital letters in Mizar anyway.) The symbol `nf` is also a used for a function, so we can not use it for an attribute `M is nf`. Instead we used `M is_nf` as a predicate synonym for this attribute. Also we can not use it for a type `nf of R`. Instead this became `Nf of R`.

6 A formal proof sketch from the Mizar mathematical library

In Section 4 we developed a Mizar formalization from a formal proof sketch. In this section we do the reverse. Here we *extract* a formal proof sketch from the proof of Newman's lemma that is in the MML:

```
 reserve R for Relation;
 reserve M,M',M'',M''',M'''',N,N' for set;
 reserve p,q for RedSequence of R;

definition
 cluster strongly-normalizing locally-confluent -> confluent Relation;
 coherence
 proof
  let R;
  assume R is strongly-normalizing;
  assume for M,M',M'' st [M,M'] in R & [M,M''] in R
   holds M',M'' are_convergent_wrt R;
  given N,N' such that
   N,N' are_divergent_wrt R & not N,N' are_convergent_wrt R;
  defpred is_confluent[set] means
   for M',M'' st R reduces $1,M' & R reduces $1,M''
    holds M',M'' are_convergent_wrt R;
  for M st for M' st [M,M'] in R & M <> M' holds is_confluent[M']
   holds is_confluent[M]
  proof
   let M;
   assume for M' st [M,M'] in R & M <> M' holds is_confluent[M'];
   let M',M'';
   assume R reduces M,M' & R reduces M,M'';
   consider p such that M = p.1 & M' = p.len p;
   consider q such that M = q.1 & M'' = q.len q;
   per cases;
   suppose len p = 1;
    thus M',M'' are_convergent_wrt R;
   suppose len q = 1;
    thus M',M'' are_convergent_wrt R;
   suppose len p <> 1 & len q <> 1;
    consider M''' such that R reduces p.2,M''' & R reduces q.2,M''';
    R reduces p.2,M';
    consider M'''' such that R reduces M',M'''' & R reduces M''',M'''';
    R reduces q.2,M'''' & R reduces q.2,M'';
    thus M',M'' are_convergent_wrt R;
  end;
  for M st M in field R holds is_confluent[M];
  thus thesis;
 end;
end;
```

The formalization from which this has been extracted occurs in article `REWRITE1`. It was written by Grzegorz Bancerek. We have renamed some variables (`a`, `b`, `c` $\rightarrow$ `M`, `M' M''`, `P` $\rightarrow$ `is_confluent`) to make this resemble the other formal proof sketch more.

The diagram that corresponds to the third '`suppose`' in this formal proof sketch is:

$$\begin{tikzcd}
& & M \arrow[dl, two heads] \arrow[dr, two heads] & & \\
& p_2 \arrow[dl, two heads] \arrow[dr, two heads] & & q_2 \arrow[dl, two heads] \arrow[dr, two heads] & \\
M' \arrow[dr, two heads] & & M''' \arrow[dl, two heads] & & M'' \arrow[ddll, two heads] \\
& M'''' \arrow[dr, two heads] & & & \\
& & {} & &
\end{tikzcd}$$

This version of the proof of Newman's lemma also uses Noetherian induction. It proves that all terms are 'confluent' under the assumption that all one-step reducts are confluent already.

7 Conclusion

7.1 Discussion

The formal proof sketch from Section 2 is quite similar to the original informal proof. Here are some deviations:

- The proof step which introduces the assumptions to the proof (`assume that R is SN and R is WCR`) and the step that states at the end that the conclusion holds (`thus thesis`) have to be present in the formal proof sketch but are left implicit in the informal proof.
- The original proof uses textual references like *this R-nf* and *such M*. In first order logic one has to quantify explicitly with the property of the referred object to get the same effect.
- The attribute `nf_inhabited` that is not present in the informal original is necessary to have a Mizar type (which has to be non-empty) corresponding to the notion *R-nf*.
- Some notation deviates from the original to avoid conflicts with the conventions of the Mizar mathematical library: `Term of R` instead of `term of R`, `Nf of R` instead of `nf of R`, `is_nf` instead of `is nf`.
- Mizar definitions can not be local to a proof. Only the `set`, `deffunc` and `defpred` constructions are local. These have a restricted syntax. Therefore we had to write `ambiguous[M]`, and could not write `M is ambiguous`.

We do not claim here that Mizar should be changed to accommodate these differences. We merely note them. We do not think that the differences are that important. (The exception might be the lack of empty types.)

7.2 Future work

As we stated in the introduction of this note, every informal proof is a challenge to write the corresponding formal proof sketch. We would like to take on more of these challenges.

It would be nice to have an experimental proof assistant that implements an automatic diagramming feature as outlined in Section 3.

Acknowledgments. Thanks to Dan Synek for coining the term 'formal proof sketch challenge'.

References

1. H. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North Holland, 1984.
2. J. Barwise and J. Etchemendy. *Hyperproof.* Number 42 in CSLI Lecture Notes. CSLI Publications, Stanford, 1995.
3. Jon Barwise and John Etchemendy. Computers, visualization, and the nature of reasoning. In T.W. Bynum and James H. Moor, editors, *The Digital Phoenix: How Computers are Changing Philosophy*, pages 93–116. Blackwell, London, 1998.
4. Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo. A Technique for Drawing Directed Graphs. *Software Engineering*, 19(3):214–230, 1993.
5. G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers.* Clarendon Press, Oxford, fourth edition, 1960.
6. M.H.A. Newman. On theories with a combinatorial definiton of "equivalence.". *Ann. of Math.*, (2):223–243, 1942.
7. F. Wiedijk. Formal proof sketches.
   URL: `<http://www.cs.kun.nl/~freek/notes/sketches.ps.gz>`, 2002.