On Making Formal Proof More Tractable

J.S. Fitzgerald, P.A. Lindsay and R. Moore

Submitted to the Formal Methods Europe Symposium 1993 (FME'93)

TECHNICAL REPORT SERIES

No. 404 October, 1992
On Making Formal Proof More Tractable

J.S. Fitzgerald, P.A. Lindsay and R. Moore

Abstract

This paper concerns the construction of formal proofs of conjectures arising in model-oriented specification. It is argued that the process of constructing a formal proof, whether by hand or with help of a mechanical prover, benefits both from a careful consideration of a rigorous form of the proof prior to formalization and from the availability of a library of useful, formally-proved results.

These points are illustrated on a simple VDM specification. It is first shown how informal argument or an attempt at the rigorous proof of a satisfiability obligation can lead to the discovery of errors in the original specification. A complete rigorous proof of the satisfiability obligation for the corrected specification is then given. The task of making this proof fully formal is then considered, and insight gained by this process is applied to simplifying the rigorous proof so that it can make use of existing results from a library of already-proven theorems.

Although the work described here has been carried out using the Vienna Development Method’s specification language and logic, the points made are by no means specific to that formalism.

© 1992 University of Newcastle upon Tyne.
Printed and published by the University of Newcastle upon Tyne, Computing Science, Claremont Tower, Claremont Road, Newcastle upon Tyne, NE1 7RU, England.
Bibliographical details

FITZGERALD, John Shaw


(University of Newcastle upon Tyne, Computing Science, Technical Report Series, no.404)

Added entries

UNIVERSITY OF NEWCASTLE UPON TYNE.

Abstract

This paper concerns the construction of formal proofs of conjectures arising in model-oriented specification. It is argued that the process of constructing a formal proof, whether by hand or with help of a mechanical prover, benefits both from a careful consideration of a rigorous form of the proof prior to formalization and from the availability of a library of useful, formally-proved results.

These points are illustrated on a simple VDM specification. It is first shown how informal argument or an attempt at the rigorous proof of a satisfiability obligation can lead to the discovery of errors in the original specification. A complete rigorous proof of the satisfiability obligation for the corrected specification is then given. The task of making this proof fully formal is then considered, and insight gained by this process is applied to simplifying the rigorous proof so that it can make use of existing results from a library of already-proven theorems.

Although the work described here has been carried out using the Vienna Development Method's specification language and logic, the points made are by no means specific to that formalism.

About the author

Dr. J.S. Fitzgerald is a Research Associate in the British Aerospace Dependable Computing Systems Centre within the Computing Science Department at the University of Newcastle upon Tyne.

Dr. P.A. Lindsay is Assistant Director of the Australian Research Council's Software Verification Centre in the University of Queensland.

Dr. R. Moore is a Research Associate on the Esprit "Afrodite" project at the Computer Science Department in the University of Manchester.

Suggested keywords

FORMAL METHODS PROOF SPECIFICATION
VALIDATION VDM VERIFICATION

Suggested classmarks (primary classmark underlined)
Dewey (18th): 511.3 001.6425 U.D.C. 510.64 519.681
On Making Formal Proof more Tractable

J. S. Fitzgerald*, University of Newcastle upon Tyne, U.K.
P. A. Lindsay, University of Queensland, Australia
R. Moore, University of Manchester, U.K.
Submitted to the Formal Methods Europe Symposium 1993 (FME’93)
September 29, 1992

Abstract

This paper concerns the construction of formal proofs of conjectures arising in model-oriented specification. It is argued that the process of constructing a formal proof, whether by hand or with the help of a mechanical prover, benefits both from a careful consideration of a rigorous form of the proof prior to formalization and from the availability of a library of useful, formally-proved results.

These points are illustrated on a simple VDM specification. It is first shown how informal argument or an attempt at the rigorous proof of a satisfiability obligation can lead to the discovery of errors in the original specification. A complete rigorous proof of the satisfiability obligation for the corrected specification is then given. The task of making this proof fully formal is then considered, and insight gained by this process is applied to simplifying the rigorous proof so that it can make use of existing results from a library of already-proved theorems.

Although the work described here has been carried out using the Vienna Development Method’s specification language and logic, the points made are by no means specific to that formalism.

1 Introduction

Formal specification is gaining in credibility as a technique which can be applied to industrial-scale development of computing systems. Part of the value of formal specification is that it admits the use of proof in checking the consistency of specifications, in validating specifications against informally-stated requirements, and in verifying design decisions through refinement.

This paper concerns practical aspects of the production of formal proofs of conjectures made about mathematical models in (model-oriented) specification and refinement. Typically such conjectures are either proof obligations or validation conjectures. Proof obligations concern specification consistency: that an expression is well-typed, that an operation has a non-empty class of implementations, or that one operation implements

*British Aerospace Dependable Computing Systems Centre, 20 Windsor Terrace, The University of Newcastle upon Tyne, Newcastle NE1 7RU, United Kingdom,
other. The specifier/developer is required to prove these. Validation conjectures are statements of properties which an informal analysis of system requirements would suggest should be true in the specification model. Proving these serves to increase confidence that the formal specification is faithful to the original informal requirements.

1.1 Informal, Formal and Rigorous Proof

The truth of a proof obligation or validation conjecture is settled by a process of argument or proof. An informal argument relies on human mathematical insight and experience. Written down, it is generally short and can be relatively easily comprehended provided the reader shares the writer’s insights. However, human insight may not be correct and a step in the reasoning of an informal argument may be discovered to be ill-founded. Care is therefore required in placing an appropriate level of confidence in the truth of an informally-proved theorem.

Use of a formal specification language with mathematically-defined semantics facilitates formal proof of the conjectures arising from a specification. A formal proof is the product of an exercise in symbol-manipulation. It can be checked by a machine, since each line in the proof represents the application of a formal rule of inference to preceding lines. The fact that a formal proof has been machine-checked increases confidence in the result which has been proved. However, formal proofs arising from realistically-sized specifications are very long and tedious for humans to construct or read, since even the most “obvious” assertions must be fully justified. This paper is concerned with improving the practicality of formal proof in the development of realistic computing systems. In particular, two techniques are of interest:

Rigorous Proof: Jones, in [Jon90], uses the term rigorous proof to refer to a proof which has been conducted to a greater level of detail than would be the case in an informal argument, so that one is confident that the proof could be made fully formal if that were required. This is also the definition adopted in [MoD91]. The Natural Deduction proof style is particularly useful here as it allows a rigorous proof to be made increasingly formal by simply filling in and expanding elided detail, as illustrated in Section 3.4 below.

This approach is adopted in the present paper, which concentrates on what might be termed the “pre-formalization” stages of proof construction. The example on which this paper is based shows that careful consideration of the rigorous proof can lead to a simplification of the task of producing the formal proof.

Collections of Proved Results: The availability of an extensible collection of already-proved theorems is a valuable asset in limiting the depth of detail to which a formal proof need be conducted. Work on the Mural proof support environment [JJLM91] outlined the basis for such a collection, through the definition of a theory store, a structured collection of axioms and proved theorems which could be used in the construction of new proofs.

[JJLM91] describes early work on producing a theory store containing a collection of formally proved results about the logic and data types of VDM. This work has been taken further, and in [BFL+93] the authors present proof rules for the logic of partial functions (LPF [BCJ84], [Che86]) and the VDM primitive data types and
type constructors, as well as illustrating how these can be used in formally proving 
conjectures arising from VDM specifications (see [BR91]).

It should be stressed that rigorous proof prior to formalization and access to collection
s of proved results are valuable whether the formal proof is constructed by hand or
using some sort of mechanical prover: some insight into how a rigorous proof might be
structured generally helps when it comes to guiding a mechanical prover (e.g. [GGH90]),
and the task of proof construction is made significantly easier if a large library of proved
results is available.

1.2 This Paper

This paper illustrates the ideas introduced above by considering the pre-formalization
stages of a proof of satisfiability of a simple operation in a small VDM specification.

The specification language used is generally that of [BS89] and [Daw91]. However,
certain abbreviations have been used and these are explained where they arise.

Theorems are stated in the logical frame of [JJLM91], again with certain simplifica-
tions for the purposes of rigorous proof which are explained where they arise.

Rigorous proofs are presented in the Natural Deduction style of [Jon90]. Lines in
each proof are justified by appealing to a particular rule of the logic (from the collection
of available proved results), by appealing to a whole collection of results, or by a short
phrase. Again, the details of this approach are described where appropriate below.

A simplified part of the specification of an aircraft collision avoidance system is con-
sidered (Section 2). A first attempt at writing an operation specification is found to be
erroneous, both through informal argument and an attempt at rigorous proof of the sat-
isfiability proof obligation (SectionS-first). The specification is corrected, and it is shown
how the rigorous proof can be completed using the abortive attempt as a starting point.

Considering the difficulties of formalizing the rigorous proof leads to insight into the
distinction between those aspects of the specification which are essential to the proof and
those which are peripheral. This leads to a new, simpler formulation of the proof which
exploits already-proved results from a library of theories (Section 4). The new rigorous
proof of satisfiability is presented in full and its (easier) formalization considered.

2 An Example Specification

The example deals with an aircraft collision avoidance system (CAS). It is inspired by
the TCAS II system now fitted to passenger aircraft over a certain size in the United
States [RTC83].

A CAS receives radar input from scans of the surrounding airspace and records data
about each other flight in the vicinity. Using this data, the projected future course of
each of these flights is calculated, and any which is predicted to come dangerously close
is designated a threat. Each threat is displayed to the pilot, together with an advisory
which suggests what evasive action should be taken. This display comprises a list of
threats and their corresponding advisories. An example display is shown in Figure 1.

The formal specification given below describes the CAS display. Each element or
record of the display is described by the type DRec, which comprises some flight identifier
<table>
<thead>
<tr>
<th>Flight Number</th>
<th>Advisory</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA5495</td>
<td>CLIMP</td>
</tr>
<tr>
<td>LH1773</td>
<td>CLIMP &lt; 2000 fpm</td>
</tr>
<tr>
<td>QA051</td>
<td>WARNING ONLY</td>
</tr>
</tbody>
</table>

Figure 1: An example warning display

of type `FlightId` and its corresponding advisory of type `Advisory`. The details of the types `FlightId` and `Advisory` are unimportant as far as this example is concerned.

```plaintext
DRec :: fid : FlightId
       adv : Advisory
```

The full display consists of a sequence of these display records. It is a requirement on the display that no flight identifier should appear twice, risking confusing the pilot. This is stated as an invariant on the display:

```plaintext
state Display of disp: DRec*
inv mk-Display(disp) ⊢ ¬ ∃i ≠ j ∈ inds disp · disp(i).fid = disp(j).fid
end
```

The "¬ ∃i ≠ j ..." is an abbreviation. The VDM-SL (as at [BSI92]) version¹ would be:

```
¬ ∃i, j ∈ inds disp · i ≠ j ∧ disp(i).fid = disp(j).fid
```

An operation `ADD` is required which adds a new record to the display. A first attempt at specifying this operation might be:

```plaintext
ADD (d: DRec)
  ext wr disp : DRec*
  pre d \∉ e elems disp
  post disp = [d] \sim disp
```

### 3 A First Attempt at a Satisfiability Proof

Showing that the operation `ADD` is satisfiable involves showing that for every input parameter and starting state which satisfy the precondition of `ADD` there is some new

¹The invariant here has been written in the style of [Daw91], believed conformant to [BSI92], i.e. as a Boolean function on elements of the composite type corresponding to the state ([Daw91], pgs. 23,25-26). In their work on proof theory for VDM specifications, the authors have come to the view that the proof theory for composite types, their constructor and selector functions, is considerably simpler if the invariant is treated as a Boolean function on values of the types of the components.
state which is related to them via the postcondition of ADD, i.e. that the operation leaves the system in a valid state provided it was already in a valid state in which it was legitimate to apply the ADD operation. This condition can be stated as an inference rule. In the shorthand notation adopted here this has the form:

\[
\begin{align*}
\text{ADD satis} & \quad \exists \text{disp} \cdot \text{disp} = [d] \# \text{disp} \land \text{inv-Display(disp)} \\
\text{inv-Display(disp)}, \\
\text{d} \notin \text{elems disp}
\end{align*}
\]

Before investing time and effort in the formal proof of an obligation or other conjecture it is wise to construct an informal argument or rigorous proof, in order to gain some conviction that the obligation is indeed provable. For the example here both an informal argument and a rigorous proof will be shown.

### 3.1 Informal argument

The proof obligation amounts to showing that adding a new display record \(d\) to an existing display \(\text{disp}\) generates a valid display provided that \(d\) is not already an element of \(\text{disp}\).

Clearly adding a display record to a sequence of display records produces a sequence of display records, but does that sequence satisfy the invariant? To do so would require that no two display records relate to the same flight. Since the starting state \(\text{disp}\) is a valid state the records in \(\text{disp}\) are known to satisfy this condition, so the only problem can be if the new record \(d\) has the same flight identifier as one of the existing records of \(\text{disp}\). The precondition of ADD ensures that \(d\) does not already occur in \(\text{disp}\). However, this can be satisfied if there is a record in \(\text{disp}\) with the same flight identifier as \(d\) but with a different advisory. In this case adding \(d\) to \(\text{disp}\) would violate the invariant in the new state, as two display records would refer to the same flight. From this, it is apparent that the precondition of the ADD operation must be strengthened to rule out the case which would result in the error.
3.2 Rigorous proof

To begin the proof, the hypotheses and conclusion are recorded. The task is to supply a justification of the conclusion, along with any intermediate steps in the reasoning in the space between the hypotheses and conclusion. The symbol "( justify )" denotes a justification to be supplied:

\[
\text{from } \text{inv-Display}(\text{disp}), \ d \notin \text{elems disp} \\
\ldots \text{infer } \exists \text{disp} \cdot \text{disp} = [d] \leadsto \text{disp} \land \text{inv-Display}(\text{disp}) \quad \langle \text{justify} \rangle
\]

Typically in such proofs, one begins by examining the conclusion and reasoning backwards. Conclusions of satisfiability proofs invariably involve existential quantifiers, and the proof thus amounts to showing that a value exists which satisfies the body of the existentially quantified expression. This generally requires the identification of a "witness value" for which the condition is true. This is relatively easy when the postcondition of the operation explicitly defines the new state in terms of the old state and input parameter(s), as in the present example. The first step in the proof then proceeds by justifying the conclusion using the "one-point rule":

\[
\begin{array}{c}
\text{one-point-rule} \\
\exists x \cdot x = a \land P(x)
\end{array}
\]

Let the witness value be called \text{disp}_0 to save writing the concatenation out in full everywhere. The fact that \text{disp}_0 is a sequence, though recorded in the informal argument, is considered obvious here and does not enter into the rigorous proof. The proof so far is therefore:

\[
\begin{array}{c}
\text{from } \text{inv-Display}(\text{disp}), \ d \notin \text{elems disp} \\
\quad \text{let } \text{disp}_0 \text{ stand for } [d] \leadsto \text{disp} \text{ throughout} \\
\ldots \quad \begin{array}{c}
\text{inv-Display(}\text{disp}_0) \\
\text{infer } \exists \text{disp} \cdot \text{disp} = [d] \leadsto \text{disp} \land \text{inv-Display}(\text{disp}) \quad \langle \text{justify} \rangle
\end{array}
\end{array}
\]

The justification of the conclusion names the inference rule applied and indicates the lines to which that rule was applied. The proof now reduces to showing that \text{disp}_0 satisfies the invariant.
from inv-Display(\(\overline{\text{disp}}\)), \(d \notin \text{elems } \overline{\text{disp}}\)

let \(\overline{\text{disp}_0}\) stand for \([d] \overset{\overline{\text{disp}}}{\sim} \overline{\text{disp}}\) throughout

\[
\begin{align*}
1 & \quad \neg \exists i \neq j \in \text{inds } \overline{\text{disp}} \cdot \overline{\text{disp}}(i).\text{fid} = \overline{\text{disp}}(j).\text{fid} \\
2 & \quad \text{from } i \neq j \in \text{inds } \overline{\text{disp}_0}, \: \text{w/log } i < j \\
\phantom{1} & \quad \textbf{(justify)} \\
\phantom{1} & \quad \textbf{(justify)} \\
3 & \quad \overline{\text{disp}_0}(i).\text{fid} \neq \overline{\text{disp}_0}(j).\text{fid} \\
4 & \quad \overline{\text{disp}_0}(i).\text{fid} = \overline{\text{disp}_0}(j).\text{fid} \\
& \quad \text{inv-Display}(\overline{\text{disp}_0}) \\
& \quad \text{infer } \exists \overline{\text{disp}} \cdot \overline{\text{disp}} = [d] \overset{\overline{\text{disp}}}{\sim} \overline{\text{disp}} \wedge \text{inv-Display}(\overline{\text{disp}}) \\
& \quad \text{one point rule (4)}
\end{align*}
\]

Figure 2: Skeleton proof of satisfiability

The invariant is a defined construct, so an appropriate next step is to expand its definition. Accordingly, line 1 above will be justified by folding the body of the definition of \(\text{inv-Display}(\overline{\text{disp}_0})\). The invariant on the old state can also be expanded. After line renumbering, this leads to the following proof\(^2\):

\[
\begin{align*}
& \quad \text{from inv-Display}(\overline{\text{disp}}), \: d \notin \text{elems } \overline{\text{disp}} \\
& \quad \text{let } \overline{\text{disp}_0} \text{ stand for } [d] \overset{\overline{\text{disp}}}{\sim} \overline{\text{disp}} \text{ throughout} \\
1 & \quad \neg \exists i \neq j \in \text{inds } \overline{\text{disp}} \cdot \overline{\text{disp}}(i).\text{fid} = \overline{\text{disp}}(j).\text{fid} \\
2 & \quad \text{(justify)} \\
3 & \quad \text{(justify)} \\
\phantom{1} & \quad \text{folded (2)} \\
\phantom{1} & \quad \text{folded (2)} \\
4 & \quad \text{inv-Display}(\overline{\text{disp}_0}) \\
& \quad \text{infer } \exists \overline{\text{disp}} \cdot \overline{\text{disp}} = [d] \overset{\overline{\text{disp}}}{\sim} \overline{\text{disp}} \wedge \text{inv-Display}(\overline{\text{disp}}) \\
& \quad \text{one point rule (3)}
\end{align*}
\]

According to line 2, it must be proved that each flight identifier appears at most once in the records in the sequence. This requires a subproof that, given any pair of distinct records of the sequence, the flight identifiers should be distinct. This suggests that the proof in its early stages would look like Figure 2, where the subproof appears at line 2. In order to complete the subproof, let \(i\) and \(j\) be distinct indices of \(\overline{\text{disp}_0}\), and without loss of generality assume \(i < j\) (line 2). Then one is required to prove that \(\overline{\text{disp}_0}(i).\text{fid} \neq \overline{\text{disp}_0}(j).\text{fid}\) (Figure 2, subproof 2).

The proof would proceed (as shown in Figure 3) by considering the cases \(i = 1\) and \(i = 1\) separately (line 2.2). The first case (subproof 2.3) corresponds to the case considered in the informal argument: since \(\overline{\text{disp}_0}(i) = d\), and since \(j > 1\) means that \(\overline{\text{disp}_0}(j) = \overline{\text{disp}}(j - 1)\) because \(\overline{\text{disp}_0} = [d] \overset{\overline{\text{disp}}}{\sim} \overline{\text{disp}}\), the check is that the new record

\(^2\)In justifications, hypotheses are numbered \(\text{mh.n.}\), for the \(n\)th hypothesis on \text{from} line \(m\). Where \(m\) is omitted, the reference is to the outermost \text{from} line of the proof.
from inv-Display(disp), d \notin \text{elems disp}

let disp_0 stand for [d] \smallfrown disp throughout

\[ \neg \exists i \neq j \in \text{inds disp} \cdot \text{disp}(i).\text{fid} = \text{disp}(j).\text{fid} \]

\[ \text{from } i \neq j \in \text{inds disp}_0, \text{ wlog } i < j \]

1.1 \quad 1 \leq i

1.2 \quad i = 1 \lor i > 1

1.3 \quad \text{from } i = 1

1.3.1 \quad \text{disp}_0(1) = d

1.3.2 \quad j > 1

1.3.3 \quad \text{disp}_0(j) = \text{disp}(j - 1)

1.3.4 \quad \text{disp}_0(j) \in \text{elems disp}

1.3.5 \quad d \neq \text{disp}_0(j)

1.3.6 \quad \text{disp}_0(i) \neq \text{disp}_0(j)

\ldots

\text{infer } \text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid} \quad \{ \text{justify} \}

2.4 \quad \text{from } i > 1

\ldots

\text{infer } \text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid} \quad \{ \text{justify} \}

\text{infer } \text{disp}(i).\text{fid} \neq \text{disp}(j).\text{fid}

3 \quad \neg \exists i \neq j \in \text{inds disp}_0 \cdot \text{disp}_0(i).\text{fid} = \text{disp}(j).\text{fid}

4 \quad \text{inv-Display(disp)_0}

\text{infer } \exists \text{disp} \cdot \text{disp} = [d] \smallfrown \text{disp} \land \text{inv-Display(disp)}

\{ \text{one point rule} \}

Figure 3: Incomplete proof of satisfiability

does not clash with any existing record. Now the precondition (hypothesis h2) states

that d \notin \text{elems disp}, and so it follows (line 2.3.6) that \text{disp}_0(i) \neq \text{disp}_0(j). At this

point in the proof one would like to conclude that \text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid}, but this

is clearly impossible from the known facts. Thus, one again comes to the realization

that the precondition is not strong enough, and the proof breaks down for exactly the

same reason as the informal argument did. Figure 3 shows the state of the (incomplete)

rigorous proof corresponding to the informal argument given so far. Note that the i > 1

case (subproof 2.4) has not yet been considered.

In a rigorous proof, the justifications do not have to be machine checked, and so need

not refer to particular inference rules. Where the author of the proof is confident about a

particular step, it may be justified by reference to the theory (collection of proven results)

from which rules would be applied to justify the step. For example, in Figure 3, line 2.1 is

justified by appealing to the theory of sequence indices, where there should be a derived

result indicating that indices are always at least 1. A more complicated example occurs

on line 2.3.6, which amounts to combined applications of the transitivity of equality and

substitution of equal values.
3.3 The corrected specification and completed proof

Both the attempts at informal argument and rigorous proof above lead to the conclusion that the precondition of the ADD operation needs to be strengthened to ensure that the flight identifier of the display record to be added is distinct from those of all existing display records. The correct version of the ADD operation is therefore given by:

\[
\begin{align*}
\text{ADD} \ (d: DRec) \\
\text{ext wr disp} : DRec^* \\
\text{pre} & \quad \exists r \in \text{elems disp} \cdot r.fid = d.fid \\
\text{post} & \quad \text{disp} = [d] \cap \text{disp}
\end{align*}
\]

Having made this change to the specification, the new attempt at the rigorous satisfiability proof can proceed from the point at which it was abandoned (Figure 3). Now the \( i = 1 \) case can be completed (Figure 4, line 2.6) because the strengthened precondition ensures that the flight identifier \( \text{disp}_0(i).fid \) does not occur in any element of \( \text{disp} \). The other case (\( i > 1 \), Figure 4, line 2.7) could have been completed with the original precondition since, by arguments similar to those used above in the abortive consideration of the \( i = 1 \) case, it only relates elements which were in the original display \( \text{disp} \), already known to satisfy the invariant.

3.4 Formalization

The formal version of the satisfiability proof obligation would contain typing assignments for all the variables. In the notation of [BFL+93] and [JJLM91] this would be written as:

\[
\begin{align*}
d: DRec, \\
\text{mk-Display}(\text{disp}): Display, \\
\exists \text{mk-Display}(\text{disp}): Display \cdot \text{disp} = [d] \cap \text{disp}
\end{align*}
\]

The rigorous proof given in Figure 4 above essentially represents a skeleton of the corresponding fully formal proof, in that the latter can be constructed from the former by expanding the shorthand notation used and filling in the missing or elided proof steps. Thus, for example, the expression

\[
\neg \exists i \neq j \in \text{inds disp} \cdot \text{disp}(i).fid = \text{disp}(j).fid
\]

represents

\[
\neg \exists i: N \cdot \exists j: N \cdot i \in \text{inds disp} \land j \in \text{inds disp} \land \neg (i = j) \land \text{disp}(i).fid = \text{disp}(j).fid
\]

in the formal notation of [BFL+93]. Hence the justification of line 3 from line 2 by the rule \( \neg \exists - \) in the rigorous proof of Figure 4 actually represents two applications of the \( \neg \exists - \) rule and one application of a form of the rule \( \neg \land - \) from the theory of propositional LPF:

\[
\begin{align*}
\delta P, \\
\frac{P \vdash \neg Q}{\neg (P \land Q)}
\end{align*}
\]
from inv-Display($\overline{\text{disp}}$), $\neg \exists r \in \text{elems } \overline{\text{disp} \cdot r} \cdot \text{fid} = \text{d} \cdot \text{fid}$

let $\text{disp}_0$ stand for $[d] \setminus \overline{\text{disp}}$ throughout

1 $\neg \exists i \neq j \in \text{inds } \overline{\text{disp} \cdot \text{disp}(i)} \cdot \text{fid} = \text{disp}(j).\text{fid}$

2 from $i \neq j \in \text{inds } \text{disp}_0$, wlog $i < j$

2.1 $1 \leq i$

2.2 $j > 1$

2.3 $\text{disp}_0(j) = \overline{\text{disp}(j - 1)}$

2.4 $\text{disp}_0(j) \in \text{elems } \overline{\text{disp}}$

2.5 $i = 1 \lor i > 1$

2.6 from $i = 1$

2.6.1 $\text{disp}_0(1) = d$

2.6.2 $\text{disp}_0(j).\text{fid} \neq d.\text{fid}$

infer $\text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid}$

2.7 from $i > 1$

2.7.1 $i - 1 \neq j - 1 \in \text{inds } \overline{\text{disp}}$

2.7.2 $\text{disp}_0(i) = \overline{\text{disp}(i - 1)}$

2.7.3 $\overline{\text{disp}(i - 1)} \cdot \text{fid} \neq \overline{\text{disp}(j - 1}).\text{fid}$

infer $\text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid}$

infer $\text{disp}_0(i).\text{fid} \neq \text{disp}_0(j).\text{fid}$

3 $\neg \exists i \neq j \in \text{inds } \overline{\text{disp} \cdot \text{disp}(i)} \cdot \text{fid} = \text{disp}(j).\text{fid}$

4 $\text{inv-Display}(\text{disp}_0)$

infer $\exists \text{disp} \cdot \overline{\text{disp} = [d] \setminus \text{disp} \land \text{inv-Display}(\text{disp})}$

unfolding (h1)

indices (2.h1)

numbers (2.h2,2.1)

def of $\text{disp}_0$, indices (2.2)

sequences (2.3)

numbers (2.1)

def of $\text{disp}_0$, indices (2.6.h1)

$\neg \exists$-E (h2,2.4)

equality (2.6.h1,2.6.1,2.6.2)

def of $\text{disp}_0$, indices (2.6.1,2.6.2.7.1)

def of $\text{disp}_0$, indices (2.7.h1)

equality (2.3,2.7.2,2.7.3)

cases (2.5,2.6,2.7)

$\neg \exists$-1 (2)

inv-Display (3)

one point rule (4)

Figure 4: Completed rigorous proof of satisfiability for ADD operation
The fully formal version of this step is shown in Figure 5.

In addition, the formal version of the one-point-rule used to justify the overall conclusion in Figure 4 has extra typing information that was not present in the rigorous version of the rule given in Section 3.2 above:

\[
\begin{array}{c}
a: A, P(a) \\
\hline
\text{one-point-rule} \\
\exists x: A \cdot x = a \land P(x)
\end{array}
\]

Thus, one is required to show that the witness value \( disp_0 \) chosen in the proof is of the correct type. This includes showing that \( disp_0 \) is of type \( DRec^* \) in addition to showing that it satisfies the invariant. Type membership in the formalism used here is taken to include satisfaction of the invariant.

Another shorthand comes with the use of the phrase "without loss of generality" in the rigorous proof (Figure 4, subproof 2). This is effectively hiding a consideration of a case distinction plus a realization that the case \( j < i \) which is omitted from the argument is entirely analogous to the case \( i < j \) considered as the formulae are symmetric in \( i \) and \( j \). This step could, however, be formalized as a generic inference rule relating such symmetric arguments. For example:

\[
\begin{array}{c}
\{i, j: N, i \neq j, P(i, j)\} \vdash_{i, j} P(j, i), \\
\hline
\text{wlog} \\
\{i, j: N, i < j\} \vdash_{i, j} P(i, j)
\end{array}
\]

In addition, the let \( \ldots \) stand for \( \ldots \) throughout notation hides some intricate reasoning about equality and typing.

However, the greatest amount of elision and use of shorthand has gone into hiding most of the manipulation of sequence indices and arithmetic which would be required to formalize the argument within subproof 2 of the rigorous proof (Figure 4). Completing all this detail would make the proof much harder to follow and may, without machine
support, diminish confidence in its correctness. However, if one thinks about the problem in more abstract terms, one realizes that the fact that the display is represented as a sequence is peripheral to the argument that no flight identifier appears more than once. This last condition implies that every record in the sequence must be distinct, which in turn suggests that the properties of the system essential to the satisfiability argument for ADD can be captured by treating the display as a set rather than a sequence, with each element of the set having a distinct flight identifier. The fact that the actual display is a sequence can then be thought of as a design decision at the user interface level.

Formally, the condition on this set of records would be:

$$\forall r_1 \neq r_2 \in \text{elems disp}_0 \cdot r_1.fid \neq r_2.fid$$

This condition is not alone sufficient, since it does not record the fact that all elements of the sequence form of the display are different. Adding a condition to this effect, namely that the sequence is non-repeating (is-unique), yields a statement equivalent to the original invariant:

$$\text{inv-equiv} \quad \text{inv-Display(disp)} \quad \text{is-unique(disp)} \land \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid$$

The rigorous proof of this conjecture is given in Appendix B below.

This transformation of the invariant allows the prover to take advantage of an existing body of proved results about non-repeating sequences. It is not necessary to know the exact definition of is-unique here, though some of its properties are given in the following (rigorous) rules:

$$\text{is-unique-def} \quad \{i \neq j \in \text{inds s}\} \vdash_{i,j} s(i) \neq s(j)$$

$$\text{is-unique-pres} \quad \text{is-unique}([a] \leftarrow s)$$

$$\text{is-unique-E} \quad \text{is-unique}([a] s), i \neq j \in \text{inds s} \quad s(i) \neq s(j)$$

4 An Alternative Proof of Satisfiability

Now consider conducting the satisfiability proof using the equivalent form of the invariant. First, the rigorous proof is presented, then its formalization is considered.

4.1 Rigorous proof

The proof begins in the same way as that of Section 3, but instead of considering the expanded definition of inv-Display(disp) as the goal to be proved, the equivalence stated above is employed, so that the proof amounts to showing that the new sequence is non-repeating (Figure 6, line 1) and that different elements of the sequence have different flight identifiers (line 3).

The full rigorous proof of satisfiability is shown in Figure 7. The fact that disp0 is non-repeating follows easily (by rule is-unique-pres) from the fact that the original sequence
from $inv\text{-}Display(disp)$, $\neg \exists r \in \text{elems } disp \cdot r.fid = d.fid$

let $disp_0$ stand for $[d]^{\neg disp}$ throughout

1 $is\text{-}unique(disp_0)$ (justify)
2 from $r_1 \neq r_2 \in \text{elems } disp_0$

... infer $r_1.fid \neq r_2.fid$ (justify)
3 $\neg \exists r_1 \neq r_2 \in \text{elems } disp_0 \cdot r_1.fid = r_2.fid$

4 $is\text{-}unique(disp_0) \land \neg \exists r_1 \neq r_2.fid \in \text{elems } disp_0 \cdot r_1.fid = r_2.fid$

5 $inv\text{-}Display(disp)$ inv-equiv (4)

infer $\exists disp \cdot disp = [d]^{\neg disp} \land inv\text{-}Display(disp)$ one point rule (5)

Figure 6: Early stages of satisfiability proof using the inv-equiv rule

$disp$ was non-repeating and the fact that $d \notin \text{elems } disp$ (Figure 7, line 2), a consequence of the strengthened precondition.

To show that flight identifiers are not duplicated, let $r_1$ and $r_2$ be two distinct elements of the display. It must be shown that they have different flight identifiers (subproof 4). There are three cases to consider (line 4.2):

1. $r_1 = d$ and $r_2$ is an element of the original display
2. $r_2 = d$ and $r_1$ is an element of the original display
3. $r_1$ and $r_2$ are distinct elements of the original display

The first and second cases (subproofs 4.3 and 4.4) follow from the precondition on the operation: namely, that the flight identifier of the new display record $d$ is different from any existing flight identifiers. The third case follows from the invariant on the original display.

Note that the rigorous proof employs a lemma (Figure 7, line 1) in showing that the fact that $d$ is not already in $disp$ follows from the operation precondition. Rigorously stated, the lemma is:

$\neg \exists r \in \text{elems } disp \cdot r.fid = d.fid$

[Lemma] $d \notin \text{elems } disp$

The lemma is simply proved by properties from the existing theories of sets and equality on values of type $DRec$. The critical rules used in its proof are:

$fid\neq-E \quad d_1.fid \neq d_2.fid \quad d_1 \neq d_2$

$\neg \exists=-E \quad \neg \exists x \in s \cdot x = a \quad a \notin s$
from \textit{inv-Display}(\overline{\text{disp}}), \neg \exists r \in \text{elems disp} \cdot r.fid = d.fid

let \text{disp}_0 \text{ stand for } [d] \cup \overline{\text{disp}} \text{ throughout}

1 \quad d \notin \text{elems disp} \quad \text{lemma (h2)}

2 \quad \text{is-unique}(\overline{\text{disp}}) \quad \land\text{-E, inv-equiv(h1)}

3 \quad \text{is-unique}(\text{disp}_0) \quad \text{is-unique-pres (1,2)}

4 \quad \text{from } r_1 \neq r_2 \in \text{elems disp}_0

4.1 \quad \text{elems disp}_0 = \text{elems disp} \cup \{d\} \quad \text{sequences, def of disp}_0

4.2 \quad (r_1 = d \land r_2 \in \text{elems disp}) \lor

\quad (r_2 = d \land r_1 \in \text{elems disp}) \lor

\quad (r_1 \neq r_2 \in \text{elems disp}) \quad \text{sets (1,4.h1,4.1)}

4.3 \quad \text{from } r_1 = d, r_2 \in \text{elems disp}

4.3.1 \quad r_2.fid \neq d.fid

\quad \text{infer } r_1.fid \neq r_2.fid \quad \neg \exists\text{-E (h2,4.3.h2)}

\quad \text{substitution (4.3.h1,4.3.1)}

4.4 \quad \text{from } r_2 = d, r_1 \in \text{elems disp}

4.4.1 \quad r_1.fid \neq d.fid

\quad \text{infer } r_1.fid \neq r_2.fid \quad \neg \exists\text{-E (h3,4.4.h2)}

\quad \text{substitution (4.4.h1,4.4.1)}

4.5 \quad \text{from } r_1 \neq r_2 \in \text{elems disp}

4.5.1 \quad \forall r_1 \neq r_2 \in \text{elems disp}_0 \cdot r_1.fid \neq r_2.fid

\quad \text{infer } r_1.fid \neq r_2.fid \quad \land\text{-E, inv-equiv(h1)}

\quad \lor\text{-E (4.5.h1,4.5.1)}

\quad \text{cases (4.2,4.3,4.4,4.5)}

5 \quad \neg \exists r_1 \neq r_2 \in \text{elems disp}_0 \cdot r_1.fid = r_2.fid \quad \neg \exists\text{-I (4)}

6 \quad \text{is-unique}(\text{disp}_0) \land \neg \exists r_1 \neq r_2.fid \in \text{elems disp}_0 \cdot r_1.fid = r_2.fid \quad \land\text{-I (3,5)}

7 \quad \text{inv-Display}(\text{disp}_0) \quad \text{inv-equiv (6)}

\quad \text{infer } \exists\text{disp} \cdot \text{disp} = [d] \cup \overline{\text{disp}} \land \text{inv-Display}(\text{disp}) \quad \text{one point rule (7)}

Figure 7: Completed satisfiability proof using the inv-equiv rule
4.2 Formalization

The process of formalization is much the same as for the sequence-based proof. However, in this case it is argued that the formalization is easier to achieve.

There is little difference in the lengths of the proofs in Figures 4 and 7, but that is a crude comparison. An examination of the justifications in the rigorous proof of Figure 7 shows that more of them appeal to individual rules and less to vague theories than the sequence-based proof of Figure 4: the formal proof of the lemma is easy to see; the two justifications by \( \wedge\text{-I}, \text{inv-equiv} \) translate easily into applications of inv-equiv followed by \( \wedge\text{-I} \); “is-unique-pres” is applied directly. Appeals to substitution in this proof amount to single applications of rules governing substitution of equal values; in the first formal proof, they corresponded to multiple applications of rules from the theory of equality. This use of particular rules means that the proof is now in a form closer to its final formalization than Figure 4.

Note the absence of justifications by appeal to the “theory of indices” in Figure 7. It may reasonably be argued that the proof itself is easier for a reader to follow, having less concrete detail about sequences.

5 Conclusions

Informal argument is sufficient to spot the simple error in the \( ADD \) operation which makes it unsatisfiable (Section 3). This emphasizes the value of checking that proof obligations are satisfied as a specification is written. The authors would go further and advocate the inclusion of validation conjectures as comments in specifications, for example that the \( ADD \) operation cannot be applied twice in succession with the same arguments. This conjecture would be embodied in the statement that the precondition of \( ADD \) is false if its postcondition is true, which would be stated as an inference rule. The rigorous version of this would have the form:

\[
\text{post } \Rightarrow \neg \text{pre } \quad \text{disp} = [d] \quad \text{\neg disp } \Rightarrow \neg (\exists r \in \text{elems } \text{disp} \cdot r.fid = d.fid)
\]

Attempting proofs of such conjectures serves as a check that the specification captures aspects of the informal requirements. This practice will be illustrated more fully in [BFL+93].

Although informal argument is sufficient to uncover the error in \( ADD \), this might not always be so. Some writers of proofs may prefer, as some of the authors do, to begin with a structured rigorous proof rather than the free-form argument.

The availability of a body of proved results has been useful in this case study. The maintenance of a theory store allows the writer of a proof to take advantage of already-defined constructs and theories: \textit{is-unique} and the theory of non-repeating sequences in the construction of the rigorous proof for the second form of the invariant, for example. Similarly, a theory about indices and sequence concatenation would aid in the construction of the sequence-based proof.

The crux of the example in the paper is the introduction of the second form of invariant in Section 3.4. This second form is appropriate only as a tool in constructing proofs about the \( ADD \) operation. It would not be appropriate to write it in the specification in place of the original, since its meaning is not as clear to the reader. It has, however, been
argued in Section 4.2 above that using the second form eases the process of formalization of the satisfiability proof.

It is evident that much of the improvement in the accessibility of the proof in the example is due to the realization that aspects of the specification (e.g. the sequential nature of the display) were more concrete than was necessary for the proof of the particular conjecture under consideration. In using the second form of the invariant, the writer of the proof is performing "selective" abstraction, considering each conjecture and proof at the appropriate level. It would not be enough to simply criticize the specification for not being sufficiently abstract in the first place, as the paper has concentrated on just one operation and one proof: in other respects the sequential nature of the display may be absolutely necessary to the specification.

It might be argued that, although the new proof may be easier to formalize and may be more gentle on the reader, a penalty has been incurred in the time and effort taken to formulate the equivalent invariant and proving the equivalence. Is it worth putting this effort into searching for a simpler formal proof when one could use a prover and tactics? In this example, what is wrong with the first attempt? After all, it leads to a perfectly sound formal proof. Proofs are not run like code and rarely read, so why spend effort improving style? In answering this, it is important to note that the example is derived from a very simple specification. When scaling up to proofs of conjectures in realistically-sized systems, the "waste" in generating more verbose and intricate formal proofs than necessary becomes significant. Secondly, arguments from the point of view of change control suggest that the more complex a proof, the harder it will be to preserve portions of it when changes are made in a specification.

Much of the formalization can be helped by tactics and provers: for example, the handling of typing assertions, the "setting up" of satisfiability proofs by one-point rules (Figures 2 and 6), and equality-based reasoning (Figure 4, conclusions to 2.6 and 2.7; Figure 7, conclusions to 4.3 and 4.4). The essential argument of the paper is that all these tools can be better employed by the proof engineer if some advance thought has gone into the proof at the level of rigorous reasoning.

For certain classes of computing system, where a very high level of confidence in the developed product is required, formal proof in the development process is considered to be worth the investment of time and skills required (e.g. [MoD91]). The authors are of the opinion that proof, rigorous at least, and possibly formal, has much wider application wherever formal specification techniques are used. It is therefore felt important to record experience of the sort illustrated in this paper, as a contribution to improving techniques and tools involved in the industrial-scale production and application of proofs.

Acknowledgements

The authors are grateful to Brian Ritchie and Juan Bicarregui of the Rutherford-Appleton Laboratory for many highly influential discussions on proof theory in VDM. They also wish to thank Professor Cliff Jones of Manchester University for his many helpful suggestions regarding the content of this paper.
References


A The example specification

\[
DRec :: fid : FlightId \\
\quad adv : Advisory \\
Display :: disp : DRec* \\
inv disp \triangleleft \exists i \neq j \in \text{inds disp} \cdot \text{disp}(i).fid = \text{disp}(j).fid
\]
ADD (d: DRec)

ext wr disp : DRec

pre \( \exists r \in \text{elems disp} \cdot r.fid = d.fid \)

post disp = \( [d] \circ disp \)

**B Proof of equivalence**

This section contains a rigorous proof of the conjecture that the new condition is equivalent to the invariant (Section 3.4):

\[ \text{inv-Equiv}(disp) \iff \text{is-unique}(disp) \land \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid \]

The conjecture is proved separately in each direction:

from \( \text{inv-Equiv}(disp) \)

1. \( \neg \exists i \neq j \in \text{inds disp} \cdot \text{disp}(i).fid = \text{disp}(j).fid \)
   unfolding (h1)

2. \( \text{from } i \neq j \in \text{inds disp} \)
   defn of DRec (2.1)

2.1 \( \text{disp}(i).fid \neq \text{disp}(j).fid } \)
   is-unique-def(2)

3. \( \text{infer } \text{disp}(i) \neq \text{disp}(j) \)

4. \( \text{from } r_1 \neq r_2 \in \text{elems disp} \)

4.1 \( \exists i_1 \neq i_2 \in \text{inds disp} \cdot \text{disp}(i_1) = r_1 \land \text{disp}(i_2) = r_2 \)
   indices (4.h1)

4.2 \( \text{from } i_1 \neq i_2 \in \text{inds disp} \cdot \text{disp}(i_1) = r_1 \land \text{disp}(i_2) = r_2 \)
   substitution (4.2.h2, 4.2.1)

4.2.1 \( \text{disp}(i_1).fid \neq \text{disp}(i_2).fid } \)

4.2.2 \( \text{infer } r_1.fid \neq r_2.fid \)
   \( \exists \cdot (4.1.4.2) \)

5. \( \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid \)
   \( \forall \cdot (4) \)

5. \( \text{infer } \text{is-unique}(disp) \land \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid \)
   \( \land \cdot (3,5) \)

from \( \text{is-unique}(disp) \land \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid \)

1. \( \text{is-unique}(disp) \)
   \( \land \cdot (h1) \)

2. \( \forall r_1 \neq r_2 \in \text{elems disp} \cdot r_1.fid \neq r_2.fid \)
   \( \land \cdot (h1) \)

3. \( \text{from } i \neq j \in \text{inds disp} \)
   is-unique-E (1.3.h1)

3.1 \( \text{disp}(i) \neq \text{disp}(j) \)
   sequences (3.1)

3.2 \( \text{disp}(i) \neq \text{disp}(j) \in \text{elems disp} \)
   \( \forall \cdot (2,3.2) \)

4. \( \neg \exists i \neq j \in \text{inds disp} \cdot \text{disp}(i).fid = \text{disp}(j).fid \)
   \( \neg \exists \cdot (3) \)

4. \( \text{infer } \text{inv-Equiv}(disp) \)
   folding (4)