Typesetting Lamport's Structured Proofs Under LaTeX

Author
Andrew Lincoln Burrow
albcorp@internode.on.net

Abstract
The AlbProofs package provides a single alb-proofs LaTeX package to implement Leslie Lamport's structured proofs (Lamport, 1993). While Lamport has provided his own package, this package is designed to rectify certain deficiencies. In particular, it provides improved type setting of labels for proof resources and proof steps, markup structure which is isomorphic to the proof's block structure, effective support for cross-referencing with AUCTeX and RefTeX, and markup in the alb name space. The package is supported by an emacs lisp file customising AUCTeX and RefTeX which constructs labels in the file namespace.

Version Information
Revision: 1.7
Date: 2006-06-23 06:33:00

1 Introduction

The `alb-proofs` LaTeX package implements Leslie Lamport's structured proofs described in (Lamport, 1993). A structured proof decomposes each argument into a sequence of steps. First come assumptions, then assertions, and finally a conclusion. Each assumption introduces a variable or condition, and each assertion is itself the goal of a nested proof. Together an assertion and its subproof are termed a *proof step*.

The purpose of the structured proof is to make all steps explicit and to sequester detail into the nested subproofs. A useful approach is to first scan the top level assertions to take in the outline, and then to examine the subproofs in detail. Ideally, the proof justifies each assertion by reference to a supporting mathematical proposition and a list of accessible assumptions and assertions. In practice, the proof is resolved until each assertion can be argued succinctly in a few short sentences.

Figure 1 shows a possible layout for a small proof. It demonstrates two levels of nesting, where there are fresh assumptions in each nesting: the outer, and the nesting under ⟨1⟩2.

It is important to understand the scope of assumptions and assertions. Access is determined by a rule that recurs on the nesting in the proof. Note that each assertion occurs in a sequence of steps with a common parent. The assertion enjoys the assumptions and assertions that precede it in the list, and also those enjoyed by its parent. For example, in Figure 1 the proof step at ⟨2⟩1 can make use of assumptions ⟨1⟩i, ⟨1⟩ii, and ⟨2⟩i and assertion ⟨1⟩1; it cannot use ⟨1⟩2 which does not precede it, but instead contains it; conversely, ⟨1⟩3 cannot use ⟨2⟩i, ⟨2⟩1, and ⟨2⟩2 but can use ⟨1⟩i, ⟨1⟩ii, ⟨1⟩1, and ⟨1⟩2 because they precede it in the only sequence in which it participates. Note that access is determined by the structure rather than the labelling.

The labelling of resources exploits the selectivity of the access rule. Since a proof step only has access to resources in one sequence per level of nesting, the sequence is uniquely identified by a nesting level. Therefore, rather than generate labels of the form 1.2.1.1, labels concatenate the nesting level, written within angle brackets, and a further counter. Assumptions use Roman numerals while assertions use Arabic numerals — this is analogous to the page numbering in a book.

The headings on assertions are extended to clarify the use of mathematical induction. In this case, the top-level assertions are marked "Assert Base" or "Assert Induction", and the conclusion is marked "Q.E.D. by Mathematical Induction". The top-level assertions are restatements of the goal narrowed for the particular case.

2 Using the Commands and Environments

The environments of `alb-proofs` place certain syntactic restrictions on their use. This section concerns these restrictions and the typographic meaning of the markup. Section 4 describes how the AUCTeX customisation eases the burden of entering syntactically correct LaTeX code.

Proof.

⟨1⟩i LET *assumption*

⟨1⟩ii LET *assumption*

⟨1⟩1 ASSERT *assertion*
Proof of the assertion.

⟨1⟩2 ASSERT *assertion*

⟨2⟩i LET *assumption*

⟨2⟩1 ASSERT *assertion*
Proof of the assertion.

⟨2⟩2 Q.E.D.
Explanation of the conclusion.

⟨1⟩3 Q.E.D.
Explanation of the conclusion.

Figure 1: An example of the syntax of a structured proof.

2.1 Structured Proof Environment

The `albProof` environment replaces the AMSLaTeX `proof` environment to support structured proofs. Content within a structured proof is recursively composed, where each level comprises an optional `albSketch` environment, an optional `albAssumptions` environment, and a mandatory `albAssertions` environment. Recursion occurs because items in `albAssertions` environments can expand into subproofs.

`\begin{albProof}`[*name*] Compose a proof for a mathematical proposition. *name* is the name of the proposition being proved. This is typically a reference to a theorem-like environment from `alb-theorems`. The environment contains an optional `albSketch` environment, an optional `albAssumptions` environment, and a mandatory `albAssertions` environment.

`\begin{albSketch}` Present a sketch of the proof. The intent is to allow comments on the proof strategy to imrpove readability.

`\begin{albAssumptions}` Enumerate the assumptions introduced at the current proof step. Each assumption is introduced by an `item` command that may be labelled. The assumption should be marked up using either `albAssume` or `albSuppose`.

`\albAssume{`*assumption*`}` Typeset an assumption in a proof. *assumption* is typically a mathematical expression. `albAssume` should be used inside `albAssumptions` to indicate assumptions unless the proof is by contradiction. For example,

```
\albAssume{$x \in \albNaturals$}
```

produces the following.

LET $x \in \mathbb{N}$

`\albSuppose{`*supposition*`}` Typeset a supposition in a proof by contradiction. *supposition* is typically a mathematical expression. `albSuppose` should be used inside `albAssumptions`. For example,

```
\albSuppose{$x \in \albNaturals$}
```

produces the following.

SUPPOSE $x \in \mathbb{N}$

`\begin{albAssertions}` Enumerate the sequence of assertions establishing the truth of a proof step. Each assertion is introduced by an `item` command that may be labelled. The assertion should be marked up using one of the commands `albAssert`, `albAssertBase`, `albAssertInduction`, `albQED`, `albQEDbyContradiction`, or `albQEDbyInduction`. These commands correspond to the various stages in a proof by cases, contradiction, or mathematical induction.

The proof of the assertion follows as either a free form description or a nested proof detailed as a `albSketch`, `albAssumptions`, and `albAssertions` sequence.

`\albAssert{`*assertion*`}` Typeset an assertion in a proof. *assertion* is typically a mathematical expression. `albAssert` should be used inside `albAssertions`. For example,

```
\albAssert{$F(x) \implies x \in \albNaturals$}
```

produces the following.

ASSERT $F(x) \implies x \in \mathbb{N}$

`\albAssertBase{`*assertion*`}` Typeset the base assertion in a proof by induction. *assertion* is typically a mathematical expression. `albAssertBase` should be used inside `albAssertions`. For example,

```
\albAssertBase{$F(0) \in \albNaturals$}
```

produces the following.

ASSERT BASE $F(0) \in \mathbb{N}$

`\albAssertInduction{`*assertion*`}` Typeset the assertion of induction in a proof by induction. *assertion* is typically a mathematical expression. `albAssertInduction` should be used inside `albAssertions`. For example,

```
\albAssertInduction{$F(n) \in \albNaturals
  \implies F(n + 1) \in \albNaturals$}
```

produces the following.

ASSERT INDUCTION $F(n) \in \mathbb{N} \implies F(n+1) \in \mathbb{N}$

`\albQED`**,** `\albQEDbyContradiction`**, and** `\albQEDbyInduction` Typeset the assertion that the sequence of proof steps is complete. The commands each terminate a sequence of proof steps according to the form of proof. For example,

```
\albQED{}
\albQEDbyContradiction{}
\albQEDbyInduction{}
```

respectively produce the following.

Q.E.D.
Q.E.D. BY CONTRADICTION
Q.E.D. BY MATHEMATICAL INDUCTION

3 AUCTeX Customisations

Under AUCTeX the file `alb-proofs.el` is automatically loaded whenever the `alb-proofs` package is used. The customisation adds support for the commands and environments from `alb-proofs`. In addition, the `reftex-label` command (`C-c (`) constructs labels for proof steps, and the `reftex-reference` command (`C-c )`) formats the context lines to better indicate the typeset output.

References

Leslie Lamport. How to write a proof. Technical report, Digital Systems Research Center, February 1993. URL `ftp://ftp.digital.com/pub/DEC/SRC/research-reports/SRC-094.ps.gz`.