THE AUSTRALIAN NATIONAL UNIVERSITY

TR-CS-96-09

Structured Calculational Proof

Ralph Back, Jim Grundy and Joakim von Wright

November 1996

Joint Computer Science Technical Report Series

Department of Computer Science
Faculty of Engineering and Information Technology

Computer Sciences Laboratory
Research School of Information Sciences and Engineering

This technical report series is published jointly by the Department of Computer Science, Faculty of Engineering and Information Technology, and the Computer Sciences Laboratory, Research School of Information Sciences and Engineering, The Australian National University.

Please direct correspondence regarding this series to:

Technical Reports
Department of Computer Science
Faculty of Engineering and Information Technology
The Australian National University
Canberra ACT 0200
Australia

or send email to:

`Technical.Reports@cs.anu.edu.au`

A list of technical reports, including some abstracts and copies of some full reports may be found at:

`http://cs.anu.edu.au/techreports/`

Recent reports in this series:

TR-CS-96-08 David Hawking and Paul Thistlewaite. *Relevance weighting using distance between term occurrences.* August 1996.

TR-CS-96-07 Andrew Tridgell, Paul Mackerras, David Sitsky, and David Walsh. *AP/Linux - initial implementation*. June 1996.

TR-CS-96-06 M. Hegland, M. H. Kahn, and M. R. Osborne. *A parallel algorithm for the reduction to tridiagonal form for eigendecomposition*. June 1996.

TR-CS-96-05 Andrew Tridgell and Paul Mackerras. *The rsync algorithm*. June 1996.

TR-CS-96-04 Peter Bailey and David Hawking. *A parallel architecture for query processing over a terabyte of text*. June 1996.

TR-CS-96-03 Brendan D. McKay. *`autoson` – a distributed batch system for unix workstation networks (version 1.3)*. March 1996.

Structured Calculational Proof

Ralph Back* Jim Grundy† Joakim von Wright‡

November 25, 1996

Abstract

We propose a new format for writing proofs, which we call *structured calculational proof*. The format is similar to the calculational style of proof already familiar to many computer scientists, but extends it by allowing large proofs to be hierarchically decomposed into smaller ones. In fact, structured calculational proof can be seen as an alternative presentation of natural deduction. Natural deduction is a well established style of reasoning which uses hierarchical decomposition to great effect, but which is traditionally expressed in a notation that is inconvenient for writing calculational proofs.

The hierarchical nature of structured calculational proofs can be used for proof browsing. We comment on how browsing can increase the value of a proof, and discuss the possibilities offered by electronic publishing for the presentation and dissemination of papers containing browsable proofs.

Note: This paper is also available as Turku Centre for Computer Science TUCS Technical Report No 65.

1 Introduction

This paper presents a new format for writing proofs, which we call *structured calculational proof*. Two of the main inspirations we have drawn on when formulating this format are natural deduction [3, 10] and calculational proof [2, 4, 13]. The clarity and readability of calculational proof has made it a popular choice among computer scientists. We feel, however, that pure calculational proof provides too little support for the formal decomposition of large proofs into smaller ones. Natural deduction, on

*Åbo Akademi University
†The Australian National University
‡Åbo Akademi University

the other hand, is well suited to the structured decomposition of proofs. However, natural deduction proofs are seldom as easy to read as calculational ones.

The structured calculational proof format proposed in this paper is an attempt to devise a system of proof presentation that combines the readability of calculational proof with the structuring facilities of natural deduction. The resulting method resembles Robinson and Staples's window inference system of hierarchical reasoning [8, 11], but maintains a visual similarity to the more pleasing notation of calculational proof.

2 Structuring Calculational Proof

In their book *Predicate Calculus and Program Semantics* [2], the authors Dijkstra and Scholten introduce and motivate the calculational proof format. They begin by making the observation that a great many proofs can be be described as a series of transformations. For example, a proof that A is equivalent to D is often decomposed into a number of intermediate steps. If three intermediate steps were used, then the proof would have the following form:

the first step establishes	$A \equiv B$	;
the second step establishes	$B \equiv C$	;
and the third step establishes	$C \equiv D$	.

Put together, these steps establish $A \equiv D$.

If the various steps of a proof are carried out independently, as depicted above, then the intermediate expressions are duplicated. If the number of steps is large, or if the intermediate expressions are themselves large, this quickly proves to be inconvenient. Dijkstra and Scholten therefore advocate using the alternative presentation below:

$$
\begin{array}{ll}
 & A \\
\equiv & \{\text{hint why } A \equiv B\} \\
 & B \\
\equiv & \{\text{hint why } B \equiv C\} \\
 & C \\
\equiv & \{\text{hint why } C \equiv D\} \\
 & D
\end{array}
$$

This, in a nut-shell, is the calculational proof format. We now propose some extensions to calculational proof that build on its original aims of brevity and convenience.

A proof that transforms a large expression can usually be decomposed into several subproofs that transform various components of the expression. For example in the proof below, both arguments of the conjunction are transformed independently.

$$
\begin{array}{ll}
& A \wedge P \\
\equiv & \{\text{hint why } P \equiv Q\} \\
& A \wedge Q \\
\equiv & \{\text{hint why } Q \equiv R\} \\
& A \wedge R \\
\equiv & \{\text{hint why } A \equiv B\} \\
& B \wedge R \\
\equiv & \{\text{hint why } B \equiv C\} \\
& C \wedge R \\
\equiv & \{\text{hint why } C \wedge R \equiv Y\} \\
& Y \\
\equiv & \{\text{hint why } Y \equiv Z\} \\
& Z
\end{array}
$$

The unchanged portion of the expression is repeated at each step of the proof. This repetition is not only tiresome to write, but is distracting to the reader. The easiest way to avoid such repetition is to consider the subproofs separately. For example, we could have presented the above proof as follows:

1.
$$
\begin{array}{ll}
& P \\
\equiv & \{\text{hint why } P \equiv Q\} \\
& Q \\
\equiv & \{\text{hint why } Q \equiv R\} \\
& R
\end{array}
$$

2.
$$
\begin{array}{ll}
& A \\
\equiv & \{\text{hint why } A \equiv B\} \\
& B \\
\equiv & \{\text{hint why } B \equiv C\} \\
& C
\end{array}
$$

3.
$$
\begin{array}{ll}
& A \wedge P \\
\equiv & \{\text{from 1}\} \\
& A \wedge R \\
\equiv & \{\text{from 2}\} \\
& C \wedge R \\
\equiv & \{\text{hint why } C \wedge R \equiv Y\} \\
& Y \\
\equiv & \{\text{hint why } Y \equiv Z\} \\
& Z
\end{array}
$$

Unfortunately, this approach introduces new barriers to readability. If we present a subproof before its result is needed, then its presentation seems unmotivated. On the other hand, if we delay presenting a subproof until after it is needed, the reader

must temporarily take its result on faith. Either way, the reader must skip back and forth across the various subproofs to gain an understanding of the proof as a whole.

It is often better to present subproofs precisely where they are needed. We do so by visually subordinating them through indentation and by marking their beginning and end with '•' and '·'. Throughout this paper we will mark the component of the expression transformed by a subproof with corners, ⌞like this⌟ before the subproof, and ⌜like this⌝ afterwards. In general however, we need only use these clarifying marks for large or complex expressions, or for expressions that contain several identical components. Using this notation, we can present the three separate calculations above as a single uniform proof.

$$
\begin{array}{ll}
 & A \wedge \llcorner P \lrcorner \\
\equiv & \{\text{transform } P\} \\
 & \quad \bullet\ P \\
 & \quad \equiv \{\text{hint why } P \equiv Q\} \\
 & \quad\ \ Q \\
 & \quad \equiv \{\text{hint why } Q \equiv R\} \\
 & \quad\ \ R \\
\cdot & \ \llcorner A \lrcorner \wedge \ulcorner R \urcorner \\
\equiv & \{\text{transform } A\} \\
 & \quad \bullet\ A \\
 & \quad \equiv \{\text{hint why } A \equiv B\} \\
 & \quad\ \ B \\
 & \quad \equiv \{\text{hint why } B \equiv C\} \\
 & \quad\ \ C \\
\cdot & \ \ulcorner C \urcorner \wedge R \\
\equiv & \{\text{hint why } C \wedge R \equiv Y\} \\
 & Y \\
\equiv & \{\text{hint why } Y \equiv Z\} \\
 & Z
\end{array}
$$

The indentation allows us to see the structure of the proof at a glance. Readers can then focus their attention on whatever aspect of the proof interests them: the overall structure of the proof, the top level of the proof, or a particular subproof.

3 Monotonicity and Subproofs

Not all calculations preserve an equivalence relationship between successive steps in the proof. For example, a calculation of the following form may be used to establish the relationship $A \Rightarrow D$.

$$
\begin{array}{l}
\quad A \\
\Rightarrow \{\text{hint why } A \Rightarrow B\} \\
\quad B \\
\equiv \{\text{hint why } B \equiv C\} \\
\quad C \\
\Rightarrow \{\text{hint why } C \Rightarrow D\} \\
\quad D
\end{array}
$$

The fact that the composition of the relations $\Rightarrow$, $\equiv$, and $\Rightarrow$ yields the relation $\Rightarrow$ is an intrinsic part of the proof. However, the composition of relations such as these is considered sufficiently self-evident that it is not presented explicitly in calculational proofs.

If a subproof is used in a calculation to establish a relationship other than equivalence, then the monotonicity properties of the expression on which the subproof is used become important. An operation f is *monotonic* with respect to the relation $<$ if for all x and y, $x < y$ implies that $f(x) < f(y)$. For example, conjunction is monotonic in its left (and right) argument with respect to implication. This follows from the fact that for all x, y, and z, $x \Rightarrow y$ implies that $x \wedge z \Rightarrow y \wedge z$. Since conjunction is monotonic with respect to implication, we can use the following structured calculational proof to establish $(A \wedge P) \Rightarrow X$.

$$
\begin{array}{l}
\quad A \wedge \llcorner P \lrcorner \\
\Rightarrow \{\text{since } \wedge \text{ is monotonic in its right argument}\} \\
\qquad \bullet\ P \\
\qquad \Rightarrow \{\text{hint why } P \Rightarrow Q\} \\
\qquad\quad Q \\
\qquad \equiv \{\text{hint why } Q \equiv R\} \\
\qquad\quad R \\
\cdot\ A \wedge \ulcorner R \urcorner \\
\equiv \{\text{hint why } (A \wedge R) \equiv X\} \\
\quad X
\end{array}
$$

There are other connectives that are not monotonic with respect to implication. Implication itself, for example, is monotonic in its right argument, but antimonotonic in its left. An operation g is *antimonotonic* with respect to the relation $<$ if for all x and y, $x < y$ implies that $g(y) < g(x)$. A structured calculation involving implication could therefore have the following form (Note the opposing directions of the implication arrows in the subproof and the surrounding proof.):

$$
\begin{array}{l}
\quad \llcorner A \lrcorner \wedge P \\
\Rightarrow \{\text{since } \Rightarrow \text{ is antimonotonic in its left argument}\}
\end{array}
$$

$$
\begin{array}{l}
\quad \bullet\ A \\
\quad \Leftarrow \{\text{hint why } A \Leftarrow B\} \\
\quad\ \ B \\
\quad \equiv \{\text{hint why } B \equiv C\} \\
\quad\ \ C \\
\\
\cdot\ \ulcorner C \urcorner \wedge P \\
\equiv \{\text{hint why } (C \wedge P) \equiv X\} \\
\quad X
\end{array}
$$

Some operations, like exclusive or, are neither monotonic nor antimonotonic. Subproofs that transform the arguments of such operations must preserve equivalence only.

4 Contextual Information

When we see an expression in context, we have more information about it than if we had seen it in isolation. This information can be helpful when transforming the expression. For example, when transforming the conclusion of an implication it is possible to assume that its antecedent is true. From now on, we will list any assumptions that follow from the context of a subproof in angled brackets at its beginning. The following calculation illustrates the use of contextual assumptions.

$$
\begin{array}{l}
\quad P \Rightarrow (\llcorner P \wedge Q \lrcorner) \\
\equiv \{\text{transform the conclusion}\} \\
\\
\quad \bullet\ \langle P \rangle \\
\quad\ \ P \wedge Q \\
\quad \equiv \{\text{from the assumption } P\} \\
\quad\ \ \mathsf{true} \wedge Q \\
\quad \equiv \{\text{propositional calculus}\} \\
\quad\ \ Q \\
\\
\cdot\ \ P \Rightarrow \ulcorner Q \urcorner
\end{array}
$$

Contextual assumptions accumulate as we descend into the subderivations of a proof. For example, it is possible to assume both P and A in the innermost subproof of the calculation below. Note that when transforming one branch of a conjunction it is possible to assume that the other branch is true.

$$
\begin{array}{l}
\quad P \wedge \llcorner Q \lrcorner \\
\equiv \{\text{transform the second conjunct}\}
\end{array}
$$

- $\langle P \rangle$
  Q
  $\equiv$ {hint why P implies $Q \equiv (A \wedge B)$}
  $A \wedge \llcorner B \lrcorner$
  $\equiv$ {transform the second conjunct}
    - $\langle A \rangle$
      B
      $\equiv$ {hint why P and A imply $B \equiv C$}
      C
  - $A \wedge \ulcorner C \urcorner$
- $P \wedge \ulcorner A \wedge C \urcorner$

Quantifiers block out contextual assumptions about the variables they bind. For example, in body of the quantification $\ldots(\forall x.\, P[x])\ldots$,[1] it is not possible to assume anything about x from the surrounding context because the x bound in the quantification is not the same as any x outside it. We will use the notation '$x/$' at the beginning of a subproof to indicate that no previous assumptions about x may be used in that proof. The calculation below uses this notation to express that the assumption $P[x]$ is unavailable in innermost proof.

$P[x] \Rightarrow (\llcorner \forall x.\, A[x] \wedge B[x] \lrcorner)$
$\equiv$ {transform the quantification}
- $\langle P[x] \rangle$
  $\forall x.\, A[x] \wedge \llcorner B[x] \lrcorner$
  $\equiv$ {transform the second conjunct}
    - $x/\langle A[x] \rangle$
      $B[x]$
      $\equiv$ {hint why $A[x]$ implies that $B[x] \equiv C[x]$}
      $C[x]$
  - $\forall x.\, A[x] \wedge \ulcorner C[x] \urcorner$
- $P[x] \Rightarrow (\ulcorner \forall x.\, A[x] \wedge C[x] \urcorner)$

5 Example

The following short example illustrates the use of structured calculational proof. The problem solved in the example is to show that union distributes over intersection, i.e.

[1] The first instance of the notation $P[x]$ stands for a formula P with possible free occurrences of the variable x. Any subsequent related instances of the form $P[y]$ stand for the formula P with the expression y substituted for all free occurences of x, with appropriate renaming of bound variables. Note also that we use parentheses to delimit the scope of quantifiers, unless the intended scope is the whole formula.

that $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$. Note that set comprehension is just another form of quantification.

$$
\begin{array}{ll}
 & A \cup (B \cap C) \\
= & \{\text{definition of set comprehension}\} \\
 & \{v \mid\llcorner v \in A \cup (B \cap C) \lrcorner\} \\
= & \{\text{transform the predicate}\} \\
 & \quad \bullet\ v \mid \\
 & \qquad v \in A \cup (B \cap C) \\
 & \qquad \equiv \{\text{definition of } \cup\} \\
 & \qquad (v \in A) \vee (v \in B \cap C) \\
 & \qquad \equiv \{\text{definition of } \cap\} \\
 & \qquad (v \in A) \vee ((v \in B) \wedge (v \in C)) \\
 & \qquad \equiv \{\text{distribute } \vee \text{ over } \wedge\} \\
 & \qquad ((v \in A) \vee (v \in B)) \wedge ((v \in A) \vee (v \in C)) \\
 & \qquad \equiv \{\text{definition of } \cup, \text{ twice}\} \\
 & \qquad (v \in A \cup B) \wedge (v \in A \cup C) \\
 & \qquad \equiv \{\text{definition of } \cap\} \\
 & \qquad v \in (A \cup B) \cap (A \cup C) \\
 & \quad \cdot\ \{v \mid\ulcorner v \in (A \cup B) \cap (A \cup C) \urcorner\} \\
= & \{\text{definition of set comprehension}\} \\
 & (A \cup B) \cap (A \cup C)
\end{array}
$$

For readers interested in a comparison, a solution to this problem using ordinary calculational proof has been presented by Gries and Schneider [5].

6 Common Proof Paradigms

Many common proof paradigms are not easily expressed in a purely calculational style. Such proofs are usually presented as a combination of calculation and informal explanation. The simplest example of this is proof by contradiction. A proof of P by contradiction can be presented as a calculation establishing $\neg P \Rightarrow \mathsf{false}$. This is sometimes accompanied by an informal explanation that the calculation is equivalent to one establishing P. By adding structure to calculational proof we are able to construct a proof by contradiction of the more direct result $P \Leftarrow \mathsf{true}$, which is equivalent to P. Other proof paradigms usually presented as an informal combination of calculation and explanation include assuming the antecedent, case analysis, mutual implication, induction, and solving simultaneous equations.

By adding structure to calculational proof we are able to present such proofs as a single formal derivation. For example, a proof by contradiction can be presented as follows.

P
$\equiv$ {by double negation}
$\neg(\llcorner \neg P \lrcorner)$
$\Leftarrow$ {transform the inner negation}

- $\neg P$
  $\Rightarrow$ {the usual proof by contradiction}
  false

· $\neg\ulcorner$false$\urcorner$
$\equiv$ {propositional calculus}
true

As a more detailed example, consider that an inductive proof of $\forall x.\, P[x]$ is usually presented as two separate calculations, one for the base case and another for the step case. These calculations are usually accompanied by an explanation that taken together they constitute an inductive proof. We can now present such a proof as a single structured calculation of the form below.

$\forall x.\, P[x]$
$\equiv$ {the principle of induction}
$\llcorner P[0] \lrcorner \wedge (\forall n.\, P[n] \Rightarrow P[n+1])$
$\Leftarrow$ {solve the base case}

- $P[0]$
  $\Leftarrow$ {the usual base case calculation}
  true

· $\ulcorner$true$\urcorner \wedge (\forall n.\, P[n] \Rightarrow \llcorner P[n+1] \lrcorner)$
$\Leftarrow$ {solve the step case}

- $n/\langle P[n] \rangle$
  $P[n+1]$
  $\Leftarrow$ {the usual step case calculation}
  true

· true $\wedge (\forall n.\, P[n] \Rightarrow \ulcorner$true$\urcorner)$
$\equiv$ {predicate calculus}
true

7 Case Study

We now present a larger example that makes use of nested calculations and contextual information to illustrate more fully the extensions we have made to the calculational proof style. The example is drawn from highschool mathematics. Our

intention here is to demonstrate that this style of reasoning is applicable not only to problems of a logical nature, but to mathematics in general.

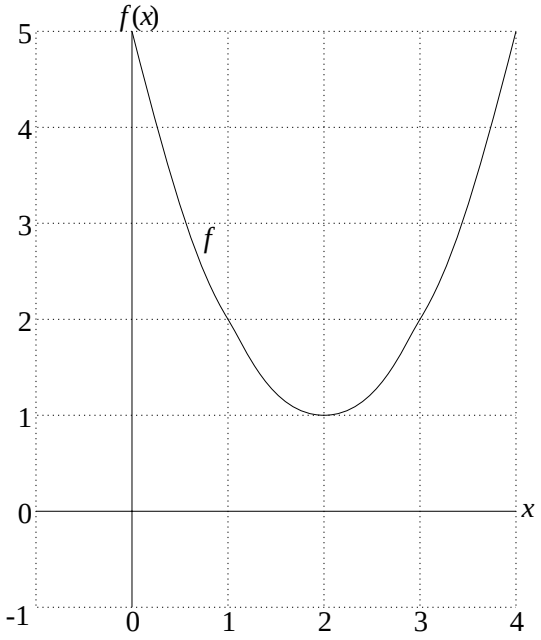

Figure 1: The Graph of Function f

Figure 1 contains the graph of a function f. Our problem is to find an expression for f as a polynomial. By inspecting the graph we can see that f is quadratic, that is $f(x)$ can be expressed as $a{\cdot}x^2 + b{\cdot}x + c$ for some a, b and c. We can also determine several points that lie on the graph, for example $f(0) = 5$, $f(1) = 2$ and $f(2) = 1$. From this information we can calculate an expression for f as follows.

$$
\begin{array}{ll}
 & (\exists a\ b\ c.\, \forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \\
 & (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \\
\equiv & \{\text{extend the scope of the existential quantifier}\} \\
 & \exists a\ b\ c.\, (\forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \\
 & \qquad \llcorner (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \lrcorner \\
\equiv & \{\text{find values for } a, b \text{ and } c \text{ that fit the points}\}
\end{array}
$$

- $a, b, c / \langle \forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c \rangle$

$$
\begin{array}{ll}
 & (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \\
\equiv & \{\text{from the assumption about } f\} \\
 & (a{\cdot}0^2 + b{\cdot}0 + c = 5) \wedge (a{\cdot}1^2 + b{\cdot}1 + c = 2) \wedge (a{\cdot}2^2 + b{\cdot}2 + c = 1) \\
\equiv & \{\text{arithmetic simplification}\} \\
 & (c = 5) \wedge \llcorner (a + b + c = 2) \wedge (4{\cdot}a + 2{\cdot}b + c = 1) \lrcorner \\
\equiv & \{\text{use the first equation to simplify the others}\}
\end{array}
$$

$\bullet$ $\langle c = 5 \rangle$
$(a + b + c = 2) \wedge (4{\cdot}a + 2{\cdot}b + c = 1)$
$\equiv$ {replace c with 5, and simplify}
$(a + b = -3) \wedge (4{\cdot}a + 2{\cdot}b = -4)$
$\equiv$ {rearrange into solutions for a and b respectively}
$(a = -3 - b) \wedge \llcorner (b = -2 - 2{\cdot}a) \lrcorner$
$\equiv$ {use the first equation to simplify the second}
$\bullet$ $\langle a = -3 - b \rangle$
$b = -2 - 2{\cdot}a$
$\equiv$ {replace a with $-3 - b$, and simplify}
$b = -4$
$\cdot$ $\llcorner (a = -3 - b) \lrcorner \wedge \ulcorner (b = -4) \urcorner$
$\equiv$ {use the second equation to simplify the first}
$\bullet$ $\langle b = -4 \rangle$
$a = -3 - b$
$\equiv$ {replace b with -4, and simplify}
$a = -1$
$\cdot$ $\ulcorner (a = 1) \urcorner \wedge (b = -4)$
$\cdot$ $(c = 5) \wedge \ulcorner (a = 1) \wedge (b = -4) \urcorner$
$\cdot$ $\exists a\ b\ c. (\forall x.\ f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \ulcorner (c = 5) \wedge (a = 1) \wedge (b = -4) \urcorner$
$\equiv$ {one-point rule[2]}
$\forall x.\ f(x) = x^2 - 4{\cdot}x + 5$

It is interesting to note that the first, and indeed, main subproof of the calculation is simply a case of solving simultaneous equations. Normally this portion of the proof would be presented separately in the distinctive style characterising the solution of simultaneous equations. In the notation we use here however, we do not need to treat such things specially, and the solution of these equations forms a uniform part of the whole proof.

8 Structured Calculation is Natural Deduction

We have described structured calculational proof as an extension of the existing calculation style of proof. We believe this new proof style to be more flexible, compact, and readable than the original. However, none of these advantages count if it is not also sound.

Calculational proofs are rarely presented completely formally, nevertheless it is important to consider the soundness of the structured calculational format. We need to establish that if an unsound proof were produced in the format, that this must

[2]The *one-point rule* for existential quantification states $(\exists x.\ P[x] \wedge (x = y)) \equiv P[y]$.

$$\frac{\Gamma_1 \vdash A \equiv B \quad \Gamma_2 \vdash B \equiv C}{\Gamma_1 \cup \Gamma_2 \vdash A \equiv C}[\equiv\equiv\text{C}] \qquad \frac{\Gamma_1 \vdash A \Leftarrow B \quad \Gamma_2 \vdash B \Leftarrow C}{\Gamma_1 \cup \Gamma_2 \vdash A \Leftarrow C}[\Leftarrow\Leftarrow\text{C}]$$

$$\frac{\Gamma_1 \vdash A \Leftarrow B \quad \Gamma_2 \vdash B \equiv C}{\Gamma_1 \cup \Gamma_2 \vdash A \Leftarrow C}[\Leftarrow\equiv\text{C}] \qquad \frac{\Gamma_1 \vdash A \equiv B \quad \Gamma_2 \vdash B \Leftarrow C}{\Gamma_1 \cup \Gamma_2 \vdash A \Leftarrow C}[\equiv\Leftarrow\text{C}]$$

Figure 2: Composition of Relations

have been the result of an erroneous proof step made by its author, and not because of the format itself. We do this by showing how the individual steps of a structured calculational proof can be reassembled to form a natural deduction proof [3, 10] of the same result. Since natural deduction is well known to be sound, we will therefore have shown that if the individual steps of the structured calculation proof are sound, then the proof as a whole is also sound.

Let us begin by considering how the individual steps of an ordinary calculational proof can be reassembled into the form of a natural deduction proof. To do this we will require a collection of derived inference rules, called *composition rules*, for composing the relations used in the proof. We will assume that the validity of the composition rules has already been established. Figure 2 contains a collection of composition rules that show how the relations $\equiv$ and $\Leftarrow$ may be composed. Note that we have opted for a sequent presentation of natural deduction so that use of assumptions is made explicit.

Consider the following short calculation making use of those relations:

$$\begin{array}{ll} & E \\ \equiv & \{\text{hint why } E \equiv F\} \\ & F \\ \Leftarrow & \{\text{hint why } F \Leftarrow G\} \\ & G \\ \equiv & \{\text{hint why } G \equiv H\} \\ & H \end{array}$$

The individual steps in this calculation can be reassembled into the following natural deduction proof using the rules from figure 2.

$$\frac{\dfrac{\vdash E \equiv F \qquad \vdash F \Leftarrow G}{\vdash E \Leftarrow G}[\equiv\Leftarrow\text{C}] \qquad \vdash G \equiv H}{\vdash E \Leftarrow H}[\Leftarrow\equiv\text{C}]$$

By inspecting both the calculation and natural deduction proof, it is easy to see how the steps of any calculational proof could be similarly reassembled.

We must now consider how to reassemble the steps of a calculation that contains subproofs into the form of a natural deduction proof. This will require an additional

collection of derived inference rules called *subproof rules*. Again we assume that the validity of the subproof rules has already been established. The most general form of a subproof is shown below, together with the corresponding general form for a subproof rule.

$E[\llcorner e \lrcorner]$
R {by transforming e}

- $x_1, \ldots x_n / \langle P_1, \ldots P_m \rangle$
  e
  r $\{\ldots\}$
  e'

$$\frac{\Gamma', P_1, \ldots P_m \vdash e \; r \; e'}{\Gamma \vdash E[e] \; R \; E[e']}$$

Γ' is those Γ with no free $x_1, \ldots x_n$

· $E[\ulcorner e' \urcorner]$

Figure 3 contains some concrete examples of subproof rules.

$$\frac{\Gamma, A \vdash B \equiv B'}{\Gamma \vdash A \wedge B \equiv A \wedge B'}[\wedge\equiv\text{2S}] \qquad \frac{\Gamma, \neg B \vdash A \Leftarrow A'}{\Gamma \vdash (A \Rightarrow B) \Rightarrow (A' \Rightarrow B)}[\Rightarrow\Rightarrow\text{1S}]$$

$$\frac{\Gamma' \vdash A[x] \Leftarrow A'[x]}{\Gamma \vdash (\forall x.\, A[x]) \Leftarrow (\forall x.\, A'[x])}[\forall\Leftarrow\text{S}] \qquad \frac{\Gamma' \vdash A[x] \equiv A'[x]}{\Gamma \vdash (\exists x.\, A[x]) \equiv (\exists x.\, A'[x])}[\exists\equiv\text{S}]$$

Γ' is those Γ with no free x $\qquad$ Γ' is those Γ with no free x

Figure 3: Subproof Rules

To reassemble the steps of a calculation with subproofs into a natural deduction proof, we begin by reassembling the innermost proofs — those that do not contain any further subproofs. The resulting natural deduction proofs may then be combined with the appropriate subproof rules and used as single steps when reassembling any enclosing proof. For example, consider the structured calculation below:

$A \wedge \llcorner B \wedge A \lrcorner$
$\equiv$ {use the first conjunct to simplify the second}

- $\langle A \rangle$
  $B \wedge A$
  $\equiv$ {use the assumption to replace A with true}
  $B \wedge \mathsf{true}$
  $\equiv$ {propositional calculus}
  B

· $A \wedge \ulcorner B \urcorner$
$\equiv$ {commutativity of $\wedge$}
$B \wedge A$

The steps of inner proof of this calculation can be rearranged into the following natural deduction proof.

$$\frac{A \vdash B \wedge A \equiv B \wedge \mathsf{true} \qquad \vdash B \wedge \mathsf{true} \equiv B}{A \vdash B \wedge A \equiv B}[\equiv\equiv\mathsf{C}]$$

We then build on this proof by applying the appropriate subproof rule.

$$\frac{\begin{array}{c}\vdots \\ A \vdash B \wedge A \equiv B\end{array}}{\vdash A \wedge B \wedge A \equiv A \wedge B}[\wedge\equiv\mathsf{2S}]$$

The resulting proof can then be used as a single step in the translation of the enclosing calculation, which can now be given as follows:

$$\frac{\begin{array}{c}\vdots \\ \vdash A \wedge B \wedge A \equiv A \wedge B\end{array} \qquad \vdash A \wedge B \equiv B \wedge A}{\vdash A \wedge B \wedge A \equiv B \wedge A}[\equiv\equiv\mathsf{C}]$$

By following the example above, it is easy to see how when given inference rules for composing relations and subproofs, it is possible to reassemble the steps of a structured calculational proof into the form of a natural deduction proof. Indeed, we can regard structured calculational proof as simply an abbreviation for a restricted form of natural deduction where those proof steps that can be inferred from the form of the proof are omitted from its presentation.

Finally, note that we allow structured calculational proofs that implicitly rely on a composition of subproof rules rather than on any individual rule. For example, consider the calculation below and its accompanying translation into natural deduction, which requires the composition of two subproof rules.

$$\begin{array}{ll} & A \wedge B \wedge \llcorner C \lrcorner \\ \equiv & \{\text{by transforming } C\} \\ & \quad \bullet\ \langle A, B\rangle \\ & \qquad C \\ & \quad \equiv \{\vdots\} \\ & \qquad C' \\ \cdot & A \wedge B \wedge \ulcorner C' \urcorner \end{array} \qquad \frac{\begin{array}{c}\vdots \\ A, B \vdash C \equiv C'\end{array}}{\dfrac{A \vdash B \wedge C \equiv B \wedge C'}{\vdash A \wedge B \wedge C \equiv A \wedge B \wedge C'}[\wedge\equiv\mathsf{2S}]}[\wedge\equiv\mathsf{2S}]$$

9 Browsing Structured Proofs

It is not usually possible, nor indeed desirable, to present a proof in complete detail. Partly this is due to the limited number of pages available to any author publishing in a journal or conference proceedings. More significantly, however, it is due to the

fact that to present any nontrivial argument in complete detail — right down to the basic axioms and inference rules of the logic — would render it unreadable.

At what level of detail should a proof be presented? If an author presents a proof in too much detail it will be difficult to read, and hence unconvincing. Including too little detail will also make a proof unconvincing. Judging the right level of detail for a proof is one of the things that distinguishes a good author. Good authors know their audience; they know what the audience will find obvious and what they will find interesting, and they present their proofs accordingly. However, since not all readers are the same, even the best author cannot present a proof in a way that is optimal for all its potential audience.

Perhaps the solution is to take the problem out of the hands of authors, and to let readers decide what details they need to see when reading a proof. A structured proof format, such as the one proposed in this paper, admits the possibility of structured browsing to increase readability. Of course, this is not possible with proofs presented on paper, but the increasing popularity of electronic publishing may soon make such considerations less important. A reader browsing a proof interactively would initially be presented with a view of the proof with all the subproofs hidden. If the reader were interested in the details of a particular subproof, they could click on the comment that describes it. The first layer of the subproof would then be revealed. In this way the reader can see not only the individual steps of a proof, but also the structure of the proof as a whole. Furthermore, the reader need only reveal as much of the proof as they find necessary to be convinced of its result.

For example, consider the proof presented in section 7. A reader interactively browsing this proof would be presented with an initial view of the proof similar to the one shown below. Here, underlining is used to indicate comments that can be expanded to reveal more detailed subproofs.

$$
\begin{array}{ll}
 & (\exists a\ b\ c.\, \forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \\
 & (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \\
\equiv & \{\text{extend the scope of the existential quantifier}\} \\
 & \exists a\ b\ c.\, (\forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \\
 & \qquad \llcorner (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \lrcorner \\
\equiv & \{\underline{\text{find values for } a, b \text{ and } c \text{ that fit the points}}\} \\
 & \exists a\ b\ c.\, (\forall x.\, f(x) = \llcorner a{\cdot}x^2 + b{\cdot}x + c \lrcorner) \wedge \ulcorner (c = 5) \wedge (a = 1) \wedge (b = -4) \urcorner \\
\equiv & \{\text{one-point rule}\} \\
 & \forall x.\, f(x) = x^2 - 4{\cdot}x + 5
\end{array}
$$

Revealing the first layer of this subproof brings us to the following view of the proof.

$$
\begin{array}{ll}
 & (\exists a\ b\ c.\, \forall x.\, f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge \\
 & (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \\
\equiv & \{\text{extend the scope of the existential quantifier}\}
\end{array}
$$

$\exists a\ b\ c.\ (\forall x.\ f(x) = a{\cdot}x^2 + b{\cdot}x + c) \wedge$
$\quad \llcorner (f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1) \lrcorner$
$\equiv$ {find values for a, b and c that fit the points}

- $a, b, c / \langle \forall x.\ f(x) = a{\cdot}x^2 + b{\cdot}x + c \rangle$
  $(f(0) = 5) \wedge (f(1) = 2) \wedge (f(2) = 1)$
  $\equiv$ {from the assumption about f}
  $(a{\cdot}0^2 + b{\cdot}0 + c = 5) \wedge (a{\cdot}1^2 + b{\cdot}1 + c = 2) \wedge (a{\cdot}2^2 + b{\cdot}2 + c = 1)$
  $\equiv$ {arithmetic}
  $(c = 5) \wedge \llcorner (a + b + c = 2) \wedge (4{\cdot}a + 2{\cdot}b + c = 1) \lrcorner$
  $\equiv$ {use the first equation to simplify the others}
  $(c = 5) \wedge \ulcorner (a = 1) \wedge (b = -4) \urcorner$

$\cdot\ \exists a\ b\ c.\ (\forall x.\ f(x) = \llcorner a{\cdot}x^2 + b{\cdot}x + c \lrcorner) \wedge \ulcorner (c = 5) \wedge (a = 1) \wedge (b = -4) \urcorner$
$\equiv$ {one-point rule}
$\forall x.\ f(x) = x^2 - 4{\cdot}x + 5$

The reader could continue to browse and expand this proof as far as they found necessary.

Ideally, authors should record their proofs in more detail than any reader could possibly want. This not only ensures the proof will be understandable to all readers, but might just help authors to notice errors in their proofs. A structured notation for proofs is also advocated by Lamport, who claims that many errors in proofs come from not carrying out proofs in sufficient detail. He recommends that authors expand their proofs until the lowest level statements are obvious, and then continue with the proof for one more level [9].

This section has tried to describe what it would be like to be able to browse proofs presented in the proposed proof format. A deeper appreciation of the possibilities offered by proof browsing can be gained from the paper *A Browsable Format for Proof Presentation* [6]. This paper appears in the electronic journal *Mathesis Universalis*, and the proofs it contains can actually be browsed in the manner described.

10 Conclusions

This paper has presented an extension to the calculational style of reasoning [2, 4, 13], which we call *structured calculational proof*. Structured calculational proof is distinguished from ordinary calculational proof by its ability to hierarchically decompose large proofs into smaller ones. The development of structured calculational proof has been driven by the first and third author's requirement for a proof notation that was both clear and compact while writing their book *Refinement Calculus: A Systematic Introduction* [1]. The book applies structured calculational proof to hundreds of problems of varying complexity. We have also used structured calculational proof to

solve an unbiased problem set in the form of the 1995 Finnish High School General Mathematics Matriculation Exam [7].

There have been three main sources of inspiration for the development of the structured calculational proof. The form of the notation is largely inspired by the original system of calculational proof [2, 4, 13]. Another major source of inspiration relates to the use of contextual information in proofs, and has been drawn from Robinson and Staples's window inference style of reasoning [11] and its subsequent generalisation and description in terms of natural deduction by Grundy [8]. The final source of inspiration has been natural deduction [3, 10], a form of reasoning with decomposition facilities which have allowed it to be used for particularly large proofs. Indeed structured calculational proof, as we have described it here, can be seen simply as a new notation for natural deduction. It is also possible to work the other way, developing structured calculational proof with natural deduction as a foundation. This approach is taken by Back and von Wright [1], leading to a more general notion of subderivation.

The addition of structuring facilities to calculational proof confers some benefits in addition to simply making it easier to transform larger expressions. Firstly, we now find that we are able to give a more formal and uniform treatment to various common proof paradigms, including proof by contradiction, assuming the antecedent, case analysis, mutual implication, induction, and solving simultaneous equations. Such arguments have traditionally been presented as an ad hoc combination of calculation and informal explanation. We also find that the structured nature of a structured calculational proof allows readers to browse a proof at various levels of detail. Direct support for such proof browsing can be supported for proofs appearing in electronic publications [6].

References

[1] Ralph-Johan R. Back and Joakim von Wright. Refinement calculus: A systematic introduction. In preperation. Turku Centre for Computer Science, Lemminkäisenkatu 14A, 20520 Turku, Finland, June 1996.

[2] Edsger W. Dijkstra and Carel S. Scholten. *Predicate Calculus and Program Semantics*, chapter 4, pages 21–29. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1990.

[3] Gerhard Gentzen. Untersuchungen über das logische Schliessen [Investigations into logical deduction]. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1935. Translated in Szabo [12], pp. 68–131.

[4] David Gries and Fred B. Schneider. *A Logical Approach to Discrete Math*, chapters 3–4, pages 41–81. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.

[5] David Gries and Fred B. Schneider. Teaching math more effectively, through the design of calculational proofs. Technical Report TR 94-1415, Department of Computer Science, Cornell University, Ithica NY 1853-7501, United States, March 1994.

[6] Jim Grundy. A browsable format for proof presentation. *Mathesis Universalis*, 1(2), Spring 1996.

[7] Jim Grundy. Structured solutions to the 1995 Finnish highschool general mathematics matriculation exam. In preperation. Turku Centre for Computer Science, Lemminkäisenkatu 14A, 20520 Turku, Finland, August 1996.

[8] Jim Grundy. Transformational hierarchical reasoning. *The Computer Journal*, 39(4):291–302, May 1996.

[9] Leslie Lamport. How to write a proof. *The American Mathematical Monthly*, 102(7):600–608, August–September 1995.

[10] Dag Prawitz. Ideas and results in proof theory. In Jens Erik Fenstad, editor, *Proceedings of the 2nd Scandinavian Logic Symposium*, volume 63 of *Studies in Logic and the Foundations of Mathematics*, pages 235–307, Oslo, 18–20 June 1970. North Holland, Amsterdam.

[11] Peter J. Robinson and John Staples. Formalizing a hierarchical structure of practical mathematical reasoning. *Journal of Logic and Computation*, 3(1):47–61, February 1993.

[12] M. E. Szabo, editor. *The Collected Papers of Gerhard Gentzen*. Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1969.

[13] Antonetta J. M. van Gasteren. *On The Shape of Mathematical Arguments*, volume 445 of *Lecture Notes in Computer Science*, chapter 14, pages 90–120. Springer-Verlag, Berlin, 1990.