

A summary of my work on Euclid’s lemma (revised)

Introduction

In this note, I summarize the work I have done on a problem which arose, in various contexts, in EWD1313 , EWD1315 , JAW0 , JAW3 , and JAW9 . This note is structured as an investigation, with heuristic commentary guiding the way.

* *
 * *

Prelude on notation

Throughout, a , b , c , x , and y denote structures of type natural. Here are the relative binding powers of our operators, from strongest to weakest:

\cdot , $\#$	function application, bag characteristic
$*$, \diamond	multiplication, unknown operator
$+$	addition
\downarrow , \Downarrow , \uparrow , \Uparrow	minimum, gcd, maximum, lcm
\leq , \sqsubseteq , $=$	at most, divides, equals
\wedge , \vee	and, or
\Rightarrow	implies
\equiv	equivalence

* *
 * *

The problem

In JAW0 , we wished to find a “nice” sufficient condition for:

$$(0) \quad a \sqsubseteq b * c \Rightarrow a \sqsubseteq b \quad ,$$

where a , b , and c are natural structures. Here, we attempt to derive such a condition using top-down analysis.

* *
 * *

Analyzing the basic shapes

The most difficult question in any investigation is where to begin. We have to find a careful balance between making progress on the one hand, and keeping things simple on the other.

At points like this, where nothing (or too many things!) come to mind, I like to embark on an analysis of the basic shapes of the manipulandum. So let's see what we can say about (0) : It is an implication, where the antecedent and consequent are syntactically similar. Both begin with $a \sqsubseteq$, the righthand argument of the antecedent is $b * c$, while the righthand argument of the consequent is subexpression b . In short, we could hardly ask for a nicer manipulandum!

As a skilled calculator, I immediately see an opportunity to exploit the transitivity of \sqsubseteq . Focussing on the arguments of $a \sqsubseteq b * c$, we observe that a and $b * c$ are syntactically very dissimilar. But we can bridge the gap between them with b : a is linked to b by expression $a \sqsubseteq b$, and b is linked to $b * c$ by being one of its subexpressions. The transitivity of \sqsubseteq allows us to bridge the gap, like so:

$$\begin{aligned}
 & a \sqsubseteq b * c \\
 \Leftarrow & \quad \{ \text{transitivity of } \sqsubseteq \} \\
 & a \sqsubseteq b \quad \wedge \quad b \sqsubseteq b * c \\
 \equiv & \quad \{ \text{a stroke of luck: property of } \sqsubseteq \} \\
 & a \sqsubseteq b \quad .
 \end{aligned}$$

How lovely: our little investigation has revealed that the converse of (0) holds everywhere, and hence we may write (0) equivalently as:

$$(0') \quad a \sqsubseteq b * c \equiv a \sqsubseteq b \quad .$$

Since there is no way to tell at this point which of (0) and (0') is the better manipulandum, we simply keep in mind that both are available to us.

As a footnote for the interested reader, the technique used above is discussed in JAW5 in a little more depth.

* *

 *

A naive solution

That being said, now that equality has entered the picture, the shape of (0') immediately suggests an application of Leibniz:

$$\begin{aligned}
 & a \sqsubseteq b * c \equiv a \sqsubseteq b \\
 \Leftarrow & \quad \{ \text{punctuality of } \sqsubseteq \} \\
 & b * c = b \\
 \equiv & \quad \{ \text{algebra } \} \\
 & b * c = b * 1
 \end{aligned}$$

$$\Leftarrow \{ \text{punctuality of } * \} \\ c = 1 .$$

So we have derived a sufficient condition, but the condition is a little too strong: it only holds for $c = 1$! Thus our solution may be deemed naive or “trivial” .

$$* \quad * \\ *$$

Towards a nicer solution: Selecting an interface

In the above calculation, we used very little about $*$ and \sqsubseteq besides their punctuality. And thus we were rewarded with a solution, $c = 1$, which has nothing to do with either of these operators. The lesson is: if we would like our solution to involve $*$ and \sqsubseteq , we had better take these into account somehow!

In order to take these operators into account, we need to use some of their properties, but which ones should we use? For the undisciplined thinker, this is a nonquestion: he has masses of knowledge about $*$ and \sqsubseteq , and will attempt to solve the problem by just “smearing” this knowledge over the problem until he stumbles across a solution. Of course, this is not our way. We would prefer to select some nice interface with our concepts, a particular way of understanding them that will hopefully cater to a clean, elegant solution. The question before us, then, is which interface to use.

In one such interface, we consider positive naturals to be unique products of primes, and then translate $*$ and \sqsubseteq in terms of this interface. This was the strategy pursued in JAW3 and JAW9 , and we will return to that strategy below, but for now we put it aside. The wholesale rewriting of a problem in one terminology in terms of another terminology is called “folding/unfolding” , and is dispreferred wherever it can be avoided. Firstly, it tends to lengthen proofs, as the foreign terminology has to be introduced, then eliminated. Secondly, because it is highly labor-intensive, it creates many opportunities for errors. And thirdly, because the original terminology is not being *used* , it indicates that the original interface is poorly chosen, or at least underdeveloped.

So instead, we pursue the direction of EWD1313 and EWD1315 , which uses the interface of lattice theory: in particular, we consider \sqsubseteq to be the partial order underlying a lattice, in which \uparrow is **lcm** and \downarrow is **gcd** .

Towards that end, we calculate with $(0')$, aiming to introduce another ingredient from lattice theory, say \downarrow :

$$a \sqsubseteq b * c \equiv a \sqsubseteq b \\ \equiv \{ \text{lattice theory} \} \\ a = a \downarrow b * c \equiv a = a \downarrow b \\ \Leftarrow \{ \text{punctuality of } = \} \\ a \downarrow b * c = a \downarrow b ,$$

and our investigation thus turns to:

$$(1) \quad a \Downarrow b * c = a \Downarrow b \quad .$$

Note that (1) is strictly stronger than (0), since under the instantiation $a, b, c := 4, 1, 2$, (0) equivaless **true**, while (1) equivaless **false**.

* * *

Exploiting a little lattice theory

Let us use our knowledge of lattice theory to see if we can simplify (1). For instance, by mutual inequality, we may write (1) equivalently as the conjunction of:

- $a \Downarrow b * c \sqsubseteq a \Downarrow b$ and
- $a \Downarrow b \sqsubseteq a \Downarrow b * c$.

As for the former, we have:

$$\begin{aligned} & a \Downarrow b * c \sqsubseteq a \Downarrow b \\ \equiv & \{ \text{lattice theory} \} \\ & a \Downarrow b * c \sqsubseteq a \quad \wedge \quad a \Downarrow b * c \sqsubseteq b \\ \equiv & \{ \text{absorption on first conjunct} \} \\ & a \Downarrow b * c \sqsubseteq b \quad . \end{aligned}$$

As for the latter, we have:

$$\begin{aligned} & a \Downarrow b \sqsubseteq a \Downarrow b * c \\ \Leftarrow & \{ \Downarrow \text{ preserves } \sqsubseteq \} \\ & b \sqsubseteq b * c \\ \equiv & \{ \text{property of } \sqsubseteq \} \\ & \mathbf{true} \quad . \end{aligned}$$

Hence (1) may be written equivalently as:

$$(1') \quad a \Downarrow b * c \sqsubseteq b \quad .$$

This is nice, but now we are stuck. Because \sqsubseteq is not a total order, we do not know how to manipulate \Downarrow on the lefthand side of \sqsubseteq . However, we write down this equivalent form of (1), as we will use it in later investigations.

* * *

A better use-interface (or calculation shape)

The approach in the last section might be best described as “flailing”. Lattice theory gave a wonderful conceptual interface for dealing with our symbols. But we took the knowledge that interface gave us and smeared it nonetheless. It was as if we were given a lovely hammer for the purpose of driving a nail, and then proceeded to use the wooden end of it! Tools are meaningless if we don’t use them well, and we are now in need of a better “use-interface” with our concepts.

In particular, we lost sight of our original observation about the shape of (1), namely the syntactic similarity between its lefthand and righthand sides. So rather than calculate with the whole expression, as we have been doing, let’s exploit this similarity by transforming one side of (1) into the other.

In which direction should we attempt this transformation? If we transform $a \Downarrow b * c$ into $a \Downarrow b$, this involves removing the symbols $*$ and c , whereas if we transform $a \Downarrow b$ into $a \Downarrow b * c$, this involves introducing the symbols $*$ and c . It seems to me that introduction is easier, thanks to the simple property of 1 with respect to multiplication:

$$(2) \quad [x = x * 1] \quad .$$

Thus we opt for the latter calculation shape, and calculate:

$$\begin{aligned}
 & a \Downarrow b \\
 = & \quad \{ \text{introducing } * \text{ via (2)} \} \\
 & a \Downarrow b * 1 \\
 = & \quad \{ \text{In order to introduce } c, \text{ we punctually assume some relation between } c \text{ and } 1. \text{ We take this relation in the form } x \diamond c = 1 \text{ for } \diamond \text{ and } x \text{ to be determined; we introduce } x \text{ so as to give us the possibility of introducing } a \text{ or } b \text{ into this relation, thereby avoiding trivial conditions like } c = 1. \text{ Note that we use the punctuality of } * \text{ and } \Downarrow \text{ here.} \} \\
 & a \Downarrow b * (x \diamond c) \\
 = & \quad \{ \text{Our goal has } b \text{ and } c \text{ as co-arguments of } * . \text{ But here one of the arguments of } * \text{ is a } \diamond\text{-term involving } c, \text{ hence distributivity is called for. Thus we require of } \diamond \text{ that } * \text{ distributes over it.} \} \\
 & a \Downarrow (b * x) \diamond (b * c) \\
 = & \quad \{ \text{Similarly, we assume distributivity of } \Downarrow \text{ over } \diamond . \} \\
 & (a \Downarrow b * x) \diamond (a \Downarrow b * c)
 \end{aligned}$$

= { We have created our goal! However, we have a bit of garbage to clean up. Let us first aim to reduce the lefthand term; we can do this by choosing $x = a$, thanks to $[a \sqsubseteq b * a]$ and the absorption property of $\Downarrow \cdot$ }

$$a \diamond (a \Downarrow b * c)$$

= { The need to eliminate the repeated a suggests we exploit idempotence, and thus we choose \Downarrow for $\diamond \cdot$ }

$$a \Downarrow b * c \quad .$$

So we have derived the condition $a \Downarrow c = 1$ and the proof of its sufficiency hand in hand, provided our assumptions in the proof about \diamond/\Downarrow are valid. These were:

- \Downarrow distributes over \Downarrow
- $*$ distributes over \Downarrow .

As for the first assumption: any associative, symmetric, and idempotent operator distributes over itself. The proof of the second assumption is beyond the scope of this note, and the interested reader is referred to EWD1315 .

Isn't it amazing what a little principled design can yield?

$$* \quad * \quad *$$

Two observations on the above

Our first observation on the above is that the heuristics are much more satisfying if we apply this same calculational strategy to the original manipulandum (0) :

$$\begin{aligned} & a \sqsubseteq b \\ \equiv & \{ (2) \} \\ & a \sqsubseteq b * 1 \\ \equiv & \{ \text{punctuality of } \sqsubseteq \text{ and } * , \text{ assuming } x \diamond c = 1 \} \\ & a \sqsubseteq b * (x \diamond c) \\ \equiv & \{ \text{assuming } * \text{ over } \diamond \} \\ & a \sqsubseteq (b * x) \diamond (b * c) \\ \equiv & \{ \text{The desire to distribute } \sqsubseteq \text{ over } \diamond \text{ immediately suggests } \Downarrow \text{ for } \\ & \quad \diamond , \text{ thanks to lattice theory.} \} \\ & a \sqsubseteq b * x \quad \wedge \quad a \sqsubseteq b * c \end{aligned}$$

$\equiv \{ \text{And now the choice } x = a \text{ is practically forced upon us. } \}$
 $a \sqsubseteq b * c \quad .$

At least for me, the choices in the last two steps are far more dictated than in the previous proof. I would like to think that this is a reflection of the fact that (0) is strictly weaker than (1) , but honestly I have no support for this claim.

Our second observation is that when the sufficient condition $a \Downarrow c = 1$ is known already, the same heuristics work even better for the design of the above calculations. (As the reader may verify.)

* *
 * *

Another interface

In the last part of this note, we retread some of the ground covered in JAW9 , for completeness, and also because I have found a way to refine the heuristics. The strategy is to start with:

(1') $a \Downarrow b * c \sqsubseteq b \quad ,$

the dead end from our previous analysis, and use the interface of unique factorization to probe further.

Every natural structure n is *pointwise* equal to a natural number, and hence can *pointwise* be associated with a unique bag containing the prime factors of n at that point. (Thus 0 may be associated with the bag of infinitely many copies of every prime.) In order to capture the information about these bags in a tidy way, we associate each n with a natural structure $\#n$, the bag characteristic. For example, since $90 = 2^1 * 3^2 * 5^1$, we have for instance:

(#90).2 = 1
 (#90).3 = 2
 (#90).5 = 1
 (#90).7 = 0 .

As we do whenever we introduce structures, we lift all relevant operators, and introduce an “everywhere operator” $\llbracket \ \ \rrbracket$ to denote universal quantification over the underlying space. In this case, the relevant operators are all operators on natural structures, and the underlying space contains the primes. So for example, $\llbracket \#x + \#y = \#z \rrbracket$ equivaless:

$\langle \forall p : p \text{ prime} : (\#x).p + (\#y).p = (\#z).p \rangle \quad .$

Of course, it would be pointless to introduce all this nomenclature if all we could do was translate back and forth between it and the more standard nomenclature. The power of the formalism lies in its own manipulative possibilities, as is revealed by the following list of properties. Letting \uparrow and \downarrow be **max** and **min**, respectively, we have for all x and y :

$$\begin{aligned}
 (3) \quad & [\quad x = y \quad \equiv \quad [\#x = \#y] \quad] \\
 & [\quad x \sqsubseteq y \quad \equiv \quad [\#x \leq \#y] \quad] \\
 & [\quad [\#(x * y) = \#x + \#y] \quad] \\
 & [\quad [\#(x \uparrow y) = \#x \uparrow \#y] \quad] \\
 & [\quad [\#(x \downarrow y) = \#x \downarrow \#y] \quad] \\
 & [\quad [\#1 = 0] \quad] \quad .
 \end{aligned}$$

Thus we have here not simply a notation, but a full-fledged calculus.

* *

 *

Calculating in the new interface

We now translate (1') using our new terminology:

$$\begin{aligned}
 & a \downarrow b * c \quad \sqsubseteq \quad b \\
 \equiv & \quad \{ (3) \} \\
 & [\quad \#(a \downarrow b * c) \leq \#b \quad] \quad .
 \end{aligned}$$

We interrupt this calculation to perform a more narrow $[]$ calculation:

$$\begin{aligned}
 & \#(a \downarrow b * c) \leq \#b \\
 \equiv & \quad \{ (3) \} \\
 & \#a \downarrow \#(b * c) \leq \#b \\
 \equiv & \quad \{ (3) \} \\
 & \#a \downarrow \#b + \#c \leq \#b \\
 \equiv & \quad \{ \text{lattice theory for total orders} \} \\
 & \#a \leq \#b \quad \vee \quad \#b + \#c \leq \#b \\
 \equiv & \quad \{ \text{algebra} \} \\
 & \#a \leq \#b \quad \vee \quad \#c \leq 0 \quad .
 \end{aligned}$$

So far, great, but (3) does not tell us how to translate a disjunction. However, since \leq is transitive, we can strengthen the disjunction of two \leq -terms into a single \leq -term, via a \leq -preserving operator. Unluckily, of our translatable operators, both \downarrow and $+$ are monotonic, so that we can proceed in two directions. First we try \downarrow , which has the potential advantage of eliminating $\#b$:

$$\begin{aligned}
 & \#a \leq \#b \quad \vee \quad \#c \leq 0 \\
 \Leftarrow & \quad \{ \downarrow \text{ preserves } \leq \} \\
 & \#a \downarrow \#c \leq \#b \downarrow 0 \\
 \equiv & \quad \{ \llbracket \#b \downarrow 0 = 0 \rrbracket \} \\
 & \#a \downarrow \#c \leq 0 \\
 \equiv & \quad \{ (3) \} \\
 & \#(a \downarrow c) \leq \#1 \quad .
 \end{aligned}$$

Excellent. Now we can return to the main calculation with (1'):

$$\begin{aligned}
 & a \downarrow b * c \sqsubseteq b \\
 \equiv & \quad \{ (3) \} \\
 & \llbracket \#(a \downarrow b * c) \leq \#b \rrbracket \\
 \Leftarrow & \quad \{ \text{the above } \llbracket \rrbracket \text{ calculation} \} \\
 & \llbracket \#(a \downarrow c) \leq \#1 \rrbracket \\
 \equiv & \quad \{ (3) \} \\
 & a \downarrow c \sqsubseteq 1 \\
 \equiv & \quad \{ \text{property of } \sqsubseteq \} \\
 & a \downarrow c = 1 \quad .
 \end{aligned}$$

That was nice, and painless too, considering we were able to capture all the relevant information about prime factorization in a tidy calculus.

As an excellent exercise, the reader may check that if we had instead exploited the fact that $+$ preserves \leq , then our calculation with (1') would have yielded:

$$\begin{aligned}
 & a \downarrow b * c \sqsubseteq b \\
 \Leftarrow & \quad \{ \text{work left to the reader} \} \\
 & a * c \sqsubseteq b \quad .
 \end{aligned}$$

At first glance, it seems as if we have discovered an alternative to Euclid’s famous lemma! But in fact this is just another too-strong “trivial” solution: Please observe that $a * c \sqsubseteq b$ implies $a \sqsubseteq b$, in which case (0) and (1) are trivially satisfied.

* *
 *

Afterword

In the beginning of this note, we performed some “flailing” lattice theory manipulations resulting in (1’), which I called a dead end under the lattice theory approach. However, a year’s pondering led me to the following beautiful calculation of (1’):

$$\begin{aligned}
 & a \Downarrow b * c \\
 \sqsubseteq & \quad \{ \text{lattice theory: } [x \sqsubseteq y \Uparrow x] \} \\
 & b \Uparrow (a \Downarrow b * c) \\
 = & \quad \{ \Uparrow \text{ over } \Downarrow, \text{ see JAW63} \} \\
 & (b \Uparrow a) \Downarrow (b \Uparrow b * c) \\
 = & \quad \{ \text{absorption, } [b \sqsubseteq b * c] \} \\
 & (b \Uparrow a) \Downarrow b * c \\
 \sqsubseteq & \quad \{ [b \Uparrow a \sqsubseteq b * a], \Downarrow \text{ monotonic} \} \\
 & b * a \Downarrow b * c \\
 = & \quad \{ * \text{ over } \Downarrow \} \\
 & b * (a \Downarrow c) \\
 = & \quad \{ \text{punctuality of } *, \text{ assuming } a \Downarrow c = 1 \} \\
 & b * 1 \\
 = & \quad \{ \text{algebra} \} \\
 & b \quad .
 \end{aligned}$$

I can’t say I have convincing heuristics for the first step, but the only other lattice theoretic manipulations of (1’) I could think of ran me in circles!

* *
 *

Postscript: One year later

JAW10 was originally written in 2005, about a week before leaving for Eindhoven. Sometime in the middle of my stay in Eindhoven, Apurva and I read through JAW10,

closely and carefully, in the manner we were accustomed to from ETAC sessions. After much struggling to understand it, Apurva's final assessment was that this could certainly be one of my finest JAWs, but that I would need to revise it thoroughly. I have finally completed my revision, and I do feel that this is indeed one of my finest JAWs. I could not have done it without his help.

One of the most striking things about this revision for me is the lack of brackets, despite the fact that all our calculations are in terms of structures, rather than scalars. This reflects my current tastes, in which the purpose of square brackets is to separate the relevant structure-valued context from the irrelevant structure-valued context. (Scalar-valued context may be shunted across the brackets.) In this investigation, we were deriving a structure-valued context in which we could calculate (0) and (1), so all our calculations were done "inside the brackets".

The experiment has been an intriguing link between past and present. I hope you enjoyed reading it as much as I enjoyed writing it.

Original: Culver City, 22 August 2005

Revision: Santa Cruz, 16 October 2006

Jeremy Weissmann
11260 Overland Ave. #21A
Culver City, CA 90230
USA
jeremy@mathmeth.com