Bill J. Ellis
August 18, 2005

# SPADEase experiment

## Contents



*Hums are short notes intended for distribution between those involved with EPSRC grant GR/T12289/01. Hums describe $\epsilon$-baked ideas, where $1 \geq \epsilon \geq 0$. (Hum refers to both the Praxis Humming bird and Winnie-the-Pooh's indirect approach to writing: "Poetry and Hums aren't things which you get, they're things which get you.").

# 1 Prologue

As indicated in Hum 15 a first version of SPADEase has been produced. Here an experiment is conducted using this system, tackling verification of the SHOLIS system. This note is a slightly revised version of the notes taken while running the experiment.

# 2 Phase 1

First, the SPADE Simplifier is executed on every SHOLIS VCG. The remaining VCs are manually inspected to identify those that might be provable. In practise this involves eliminating those that are obviously not provable. All of these potentially provable VCs are passed to SPADEase and the results are noted below.

## 2.1 basictypes\round.vcg

Zero. Uses div. Provability is dubious - not obviously false but seems so.

## 2.2 basictypes\sqrt.vcg

Zero. Some of these VCs involve div, and are difficult to tackle. However, some seem like more progress should have been achieved? Is there a bug in the decomposition method?

## 2.3 coc1p\incrementlogtens.vcg

Zero. Uses div. Not sure how to begin to solve this. Needs new heuristic?

## 2.4 cursel\vectorinlimits.vcg

Zero. Uses div. Again, not sure how to tackle such occurrences of div.

## 2.5 db\convertdbsholtoshol\setlimitregionnull.vcg

Zero. (With out of memory error also) Universally quantified conclusion post condition VC not considered by heuristics. Size causes error in code, but this is probably avoidable with alternative code.

## 2.6 db\convertdbsholtoshol\setline\lineargradient.vcg

Zero. Uses divide (/). Heuristics for this are possible, but do not exist yet.

## 2.7 display\updatepage.vcg

Zero. (With out of memory error also) Proof via cases seem likely. Need new cases heuristic. But forming this seems intuitive.

## 2.8 emerp\updatedynamicscreen.vcg

Zero. (With out of memory error also) Proof via cases seem likely. Need new cases heuristic.

## 2.9 graphics\clearpage.vcg

Zero. Fiddly working with enumerated types. Need improved support - but forming this seems intuitive.

## 2.10 graphics\copypagearea.vcg

Zero. Fiddly working with enumerated types. Need improved support - but forming this seems intuitive.

## 2.11 graphics\draw360degreecone.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.12 graphics\drawconelines.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.13 graphics\drawconelines\drawconearc.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.14 graphics\drawsinglecone.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.15 graphics\findcharindex.vcg

Prove two conclusions. These are very trivial. Recent changes to the SPADE Simplifier prevents the use of some rules in the rls file (those with variables), breaking some previous proofs.

### 2.16 graphics\insertnumber.vcg

Zero. (Odd bug?)

### 2.17 graphics\insertstring.vcg

Zero. (Odd bug? - as above)

### 2.18 graphics\plotrolllimit.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.19 graphics\selectwritingpage.vcg

Zero. Introduce enum position bounds from enum bounds required.

### 2.20 graphics\showcoc1pageheading.vcg

Zero. Uses div. Not sure how to tackle such occurrences of div.

### 2.21 graphics\showcoc4pageheading.vcg

Zero. Uses div and /. Many of these do not seem provable (about two conclusions seem plausible - but use div).

### 2.22 graphics\showcurrentshol.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.23 graphics\showpage.vcg

Zero. Introduce enum position bounds from enum bounds required.

### 2.24 graphics\showpitchgraphic\updatepitchbargraph.vcg

Zero. Uses divide (/). Heuristics for this are possible, but do not exist yet.

### 2.25 graphics\showpitchlimit\drawpitchlimitline.vcg

Zero. Uses divide (/). The other VCs do not seem provable. The divide VCs may not be provable.

### 2.26 graphics\showrelwinddir.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.27 graphics\showrelwindlabel.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.28 graphics\showrelwindspeed.vcg

Zero. Proof via cases seem likely. Need new cases heuristic.

### 2.29 graphics\showrolllimit\drawrolllimitline.vcg

Zero. Uses divide (/).

### 2.30 polarmath\polaraddtrue.vcg

Zero. Uses divide (/). Only a few VCs actually seem provable.

### 2.31 sensor\averagerelwinddirection.vcg

Zero. Uses mod. Heuristics may be possible. Case seems uncommon.

### 2.32 sensor\averagetruewinddirection.vcg

Zero. Uses mod. Heuristics may be possible. Case seems uncommon.

### 2.33 sensor\updatetruewind.vcg

Zero. Uses div - in the context of record access. Heuristics need to deal with spark record access and using div. Seems difficult.

### 2.34 sio5\copydisplaybuffertoslot5sio.vcg

Two conclusions. Those proved are hard invariant problems. Another VC may be provable with a deeper search. The rest do not seem provable.

### 2.35 sio5\haltdisplayoutput.vcg

Zero. (With out of memory error also) Partial correctness array properties in the context of many hypotheses.

## 3 Observations

Key problems encountered:

- **div (and mod)** - Integer division is generally difficult to work with as it does not describe a linear function. In particular VCs involving a single occurrence of div, or div outside the context of multiplication, are difficult to reduce as suitable rewrite rules can not logically exist. It seems that the key to tackling div is a new kind of rule that allows exploring the range of a division calculation rather than its actual value. This would be difficult to formulate. Note that the problems encountered for div are true for mod also.
- **divide (/)** - Unlike div, divide is a linear function, and more powerful rules are available. However, reasoning with divide is rare, and difficult, and not been considered by SPADEase.
- **cases** - In a few VCs it seems that a proof by cases might be key to completing the proof. While implementing a cases heuristic would be involved, it does seem quite achievable.
- **enum** - Enumerated types are problematic as they are often used, via functions, as integer numbers. This tends to create expressions whose meaning depends on external rules and hypotheses. Untangling this structure to reveal the semantics is fiddly.

- **bugs -** Out of memory errors occur in the displaying of goals during planning. The current display predicate creates a single term for the display of hypothesises, and this exceeds the maximum size supported by Sicstus. This can be fixed. However, it only occurred in a few VCs, for which the existing heuristics are unlikely to apply anyway. Some unusual behaviour and clear bugs were found. Namely, the odd failure of the decomposition method, and an attempt at evaluating the value of an atom in the elementary method.

Some key points:

- VCs are not being proved as they contain complicated constructs for which proof (let alone automated proof) is difficult to consider.
- Unprovable SHOLIS VCs have already been 'mined' by the SPADE Simplifier development - thus those that still remain are very difficult indeed and do not exhibit any clear patterns.
- The SPADE Simplifier appears to lack strategies for case splits and working with enumerated values. These are good areas to explore for any future SPADEase heuristics.
- As SPADEase does not support program analysis it is less effective than NuSPADE where additional annotations are key to completing a proof.
- It would probably have taken more than a week to have conducted this analysis using NuSPADE. SPADEase behaved (surprisingly) robustly at tackling and reporting its success on these difficult VCs. In terms of the original objectives for an industrially strong proof planner SPADEase performed well.

# 4 Phase 2

The bugs encountered during phase 1 are eliminated. The affected examples are repeated to determine the true SPADEase results.

## 4.1 basictypes\sqrt.vcg

Zero. The decomposition does not fire as a consequence of the intended behaviour of the decomposition method. Refinements of the method have to balance the cost of search with the generality of decomposition. A fully uninstantiated RHS decomposition method was considered. This did successfully decompose the problem - to unprovable goals. It seems the goals deemed provable are not actually.

## 4.2 graphics\insertnumber.vcg

Six conclusions. Atoms were evaluated as integers. Adding a check for integers fixes this. It seems all of the provable conclusions were proved - however these are trivial rewrites, currently not got by the latest SPADE Simplifier.

## 4.3 graphics\insertstring.vcg

Four conclusions. Same nature as above.

## 4.4 sio5\copydisplaybuffertoslot5sio.vcg

Three conclusions. The maximum depth searched was increased from 25 to 35. As proof planning is large grain, the depth of the search required is expected to grow at a slower rate than the goals complexity. These VCs are particularly complex and involved. The depth explored could be set depending on the complexity of the goal, or even just the nature of the goal. A fixed value is not a particular limitation however.

# 5 Phase 3

A method might be 'throw together' to exploit the enum pattern seen in a few examples. This would demonstrate the flexibility in extending SPADEase in reaction to common failure patterns. Significant progress was made on this extension in a short period of time (hours). However, it became apparent that a realistic solution would be slightly more involved than first appears. The changes requires the identification of enumeration functions, searching through conditional rewrite rules, and fitting this within the planning strategy in an efficient manner. It is guessed, following some experimentation in the SPADE Proof Checker, that this proposed heuristic would help advance, and very possibly complete the following examples.

- graphics\clearpage.vcg
- graphics\copypagearea.vcg
- graphics\selectwritingpage.vcg
- graphics\showpage.vcg

# 6 Conclusions

The original version of this note did not make any conclusions[1]. It is evident that the number of additional proofs (10) is rather small given the scale of SHOLIS. However, as the SHOLIS examples have been used to guide SPADE Simplifier development, the SPADE Simplifier leaves very little for SPADEase to tackle. In particular, the examples left unprovable by SPADEase are quite intractable. These strange combinations of various functions are difficult to tackle even using interactive proof. Further, the SHOLIS system is not heavily annotated. There are many unprovable VCs that a full system, with program analysis, might make progress with.

Thus, the above experiment serves to demonstrate that SPADEase struggles at proving really difficult conjectures. It also highlights the valuable leverage that program analysis has to offer in automated program reasoning. Further, the ability for SPADEase to robustly tackle these various problems (albeit, typically, not finding a complete proof plan) demonstrates how its fundamental ideas can indeed be realised in an industrial context.

A consequence of the experiment was to identify a small class of problems that might be solved using new enumeration and case split heuristics. As proof planning is built around coherent heuristic components its failure, in terms on these coherent components, can be illuminating. The use of critics to automatically introspect and patch on failed proof plans is well understood. However, in the absence of automated critics, the user can act as a manual critic, examining patterns of proof failure to identify common problems and invent new heuristics. This idea of using proof planning to assist in the development of proof planning is natural and not novel, however it is another feature of SPADEase with merit.

[1] I was, instead, busy attending the Praxis staff meeting, *i.e.* dinning, on a boat, somewhere in Oxford…