CS-Note* 82

Bill J. Ellis
October 29, 2004

Praxis trip III: System T

1 Introduction

Evaluates the result of program analysis on system T. There are a few locations in system T that warrant the addition of annotations.

2 Subprograms considered

2.1 admin.opisavailable

NoAnnotation-Before	NoAnnotation-After	Annotation-Before	Annotation-After
30	4	31	4

2.2 bio.getreturncode

Note that this example was incorrectly considered. Although a useful property could be found at an invariant, this is not required for completing its proof.

NoAnnotation-Before	NoAnnotation-After	Annotation-Before	Annotation-After
6	0	6	0

2.3 display.combinelines

NoAnnotation-Before	NoAnnotation-After	Annotation-Before	Annotation-After
26	3	26	2

2.4 keystore.digest

NoAnnotation-Before	NoAnnotation-After	Annotation-Before	Annotation-After
341	18	341	5

2.5 tokenreader.init

NoAnnotation-Before	NoAnnotation-After	Annotation-Before	Annotation-After
50	2	50	2

3 Truthfully lying with statistics

NuSPADE tackles those difficult examples that are currently not targeted by the Praxis tools. These examples are few and far between, and, with the growing strength of the Praxis tools, gradually diminishing.

If considering only selected examples our statistics should happily reveal that we can prove 100% of the examples that we target. This is not especially enlightening. Comparing the ratio of generated VCs to proved VCs after the Simplifier and after NuSPADE would be fairer. Here the interesting examples are quite reasonably classified as those that are currently not automatically proved. Alas, the number of trivial, lightweight, automatically proved VCs vastly overshadows the difficult VCs. Demonstrating a percentage point improvement (1%!) would be an encouraging result, however it would be rather difficult to build a convincing argument around this: 1% is still only 1%. Using actual numbers and underlining that these are the difficult problems may assist the argument.

If the Praxis tools did not add type information these statistics would be more flattering. Academically, adding type information is not especially exciting, yet the stronger figures would be pleasing. Currently the Praxis tools do not report VCs that are false. From my experiences with system S and T, this is actually quite a common cause

*Notes in this series are for ϵ-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited or distributed outwith the NuSPADE project (EPSRC GR/R24081).

for undischarged VCs. This is deemed an area where some larger numbers might be gathered with relatively little effort.

Note, from my second trip to Praxis a collection of provable VCs was gathered from system S and T. The numerous and patently uninteresting[1] unprovable VCs were not collected. Of course, this is no longer the case. It may be advantageous to negotiate with Praxis some means to explore these cases also. Although fairly numerous, they present a small, fractured coverage of the VCs which themselves are quite distant from the original code.

3.1 Compare within targeted subprograms

NoAnnotation-Before	**NoAnnotation-After**	**NoAnnotation-Proved (%)**
453	27	94.0397351
Annotation-Before	**Annotation-After**	**Annotation-Proved (%)**
454	13	97.13656388

3.2 Compare within unproved VCs

Adjust for individual subprograms being considered, by subtracting the VCs belonging to the considered subprogram from the overall number of VCs.

Before Adjust	$608 + 1461 = 2069$	172
Adjust: admin.opisavailable	$-(0 + 30)$	-2
Adjust: bio.getreturncode	$-(0 + 6)$	0
Adjust: display.combinelines	$-(3 + 17)$	-6
Adjust: keystore.digest	$-(88 + 247)$	-6
Adjust: tokenreader.init	$-(2 + 42)$	-6
After Adjust	1634	152

And consider the targeted examples in the context of the entire system collection of undischarged VCs.

NoAnnotation-Before	**NoAnnotation-After**	**NoAnnotation-Proved (%)**
$1634 + 453 = 2087$	$152 + 27 = 179$	91.42309535
Annotation-Before	**Annotation-After**	**Annotation-Proved (%)**
$1634 + 454 = 2088$	$152 + 13 = 165$	92.09770115

[1]...