Selecting the Domain of a Standard Library for a Mathematical Proof Checker

Freek Wiedijk
Nijmegen University
<freek@cs.kun.nl>

Abstract

By investigating the programs of the mathematics studies in the six main Dutch universities, we determine the basic subject areas that should be present in a standard library for a mathematical proof checker. An approach to realize such a library is outlined.

1 Introduction

The programming language C has what is called its *standard library.* It consists of a collection of basic software routines that's considered to be of general use to all users of the language. Likewise, one would expect a mathematical *proof checker* (a program meant to check a computer encoding of mathematical proof for its validity, like for instance Mizar, or the various LF-style tactic provers, like Coq) to have such a standard library. It ideally should contain the basic developments of the generally-known subject areas of mathematics.

The approach that this paper takes to determine which subjects should be in such a library is to consider a proof checker to be something like a student of mathematics. We claim that the mathematics that every student of mathematics knows is exactly what should be present in such a library. If we take this image literally we should start with the mathematics one learns in primary school ('arithmetic') and in secondary school ('first order reasoning' and 'formula manipulation'.) However, those skills are probably that basic that they should be 'built-in' to the system, and not be part of an external library. That leaves us with university level subjects. It seems reasonable to put in a standard library those subjects that *every* student of mathematics learns.

If we look at the Dutch university study programs, it turns out that in all six universities for the first two years all students follow the same program (after one year they already get their 'propedeuse' degree, but that's an artifact of the

Dutch educational system) and after that they specialize and study divergent topics. So the subjects from those first two years is what we claim should be in a mathematical standard library. In order to determine what exactly these subjects are we compared the programs of six Dutch universities (here referred to with their abbreviations: UVA, VU, UU, RUL, RUG and KUN.) It turns out that they all teach almost exactly the same topics (only with some variation in how they are named.) This paper lists those topics.

2 Subjects

The subjects taught in the first two years of mathematics in a Dutch university fall in four categories:

- Pure mathematics, broad subjects: 3 subjects
- Pure mathematics, specific subjects: 9 subjects
- Applied mathematics, mathematical subjects: 4 subjects
- Applied mathematics, other disciplines: 2 subjects

Apart from these four kinds, often a few non-mathematical subjects are taught (like the relation between mathematics and society), and also there generally are a few 'encyclopedic' subjects that give an overview of the whole field without being about anything specific.

So here is the list of subjects[1], with for each subject the MSC-classification (if appropriate), the name (both in English and in Dutch), and an indication how 'big' the subject is (the average number of courses the subject gets):

[1] For reference, here are the variants of this program for each of the six universities. For each subject we give the number of courses about it at that university and the way it is named there (which is in Dutch, of course).

UVA: a:3/Algebra, b:3/Analyse, c:2/Calculus, d:1/Discrete wiskunde, e:1/Functietheorie, f:0, g:1/Meetkunde, h:0, i:2/Lineaire algebra, j:1/Logica, k:1/Kansrekening, l:1/Topologie, m:1/Numerieke wiskunde, n:0, o:2/Statistiek, p:1/Simuleren en modelleren, q:2/Programmeren, r:3/Natuurkunde/Sterrenkunde; Other subjects: 1/Dynamische systemen, 1/Taal der Wiskunde, 1/Wetenschap en samenleving. Total: 28 courses.

VU: a:2/Algebra, b:2/Analyse, c:2/Calculus, d:2/Grafentheorie/Discrete Wiskunde, e:1/Complexe-functietheorie, f:0, g:1/Meetkunde, h:0, i:2/Lineaire Algebra, j:1/Logica, k:2/Waarsch.rek., l:2/Topologie, m:1/Numerieke Wiskunde, n:1/Besliskunde, o:2/Statistiek, p:0, q:2/Programmeren/Datastructuren, r:1/Mechanica; Other subjects: 1/Algemene Vorming, 1/Basisbegrip. Wisk., 1/Encyclopedie, 1/Maple, 1/Werkgroep Wiskunde, 2/Wiskunde Werkt. Total: 31 courses.

UU: a:4/Algebra, b:4/Analyse, c:2/Infinitesimaalrekening, d:0, e:0, f:0, g:1/Meetkunde, h:1/Maat en integratie, i:0, j:0, k:1/Kansrekening, l:1/Topologie, m:1/Numerieke wiskunde, n:0, o:1/Stochastiek, p:1/Modellen en computers, q:1/Computergebruik, r:0; Other subjects: 2/Kaleidoscoop, 1/Overdragen van wiskunde. Total: 21 courses.

RUL: a:3/Algebra, b:4/Analyse, c:0, d:1/Discrete wiskunde, e:0, f:0, g:1/Meetkunde, h:0, i:1/Lineaire algebra, j:0, k/p:2/Kansrekening en statistiek, l:1/Topologie, m:1/Numerieke wiskunde, n:1/Besliskunde, o:0, p:2/Modelleren, q:1/Programmeermethoden, r:0; Other subjects: 1/Caleidoscoop. Total: 19 courses.

		Subject	*Dutch name*	*size*
a		Algebra	Algebra	3.0
b		Analysis	Analyse	3.0
c		Calculus	Calculus	1.0
d	05	Combinatorics	Discrete wiskunde	1.0
e	30	Complex Variables	Functietheorie	0.5
f	34	Differential Equations	Differentiaalvergelijkingen	0.5
g	51	Geometry	Meetkunde	1.0
h	28	Integration	Integraalrekening	0.5
i	15	Linear Algebra	Lineaire algebra	1.0
j	03	Mathematical Logic	Logica	0.5
k	60	Probability Theory	Kansrekening	1.0
l	54	Topology	Topologie	1.0
m	65	Numerical Analysis	Numerieke wiskunde	1.0
n	90	Operations Research	Besliskunde	0.5
o	62	Statistics	Statistiek/Stochastiek	1.5
p	93	Systems Theory	Modelleren	1.0
q	68	Computer Science	Programmeren/Computergebruik	2.0
r		Physics	Fysica/Mechanica	1.5
				21.5

The subject matter of most of the items in this list will be clear. The difference between Analysis (b) and Calculus (c) is subtle: both seem to be about the behavior of real valued functions. However, the kind of mathematical activity (in the first case the goal is to develop the fundamental theory, while in the second case it's about proving and applying 'calculation rules') is quite different. Therefore we claim they're not the same subject after all, and deserve separate entries in the list. Similarly Differential Equations (f) and Integration (h) are 'substantial' enough on their own to merit their own entry.

3 Proposal

The best way to create a first version of a mathematical standard library like it's proposed here, seems to be the following. For each subject in this list a

RUG: a:3/Algebra, b:2/Analyse, c:0, d:0, e:1/Functietheorie, f:1/Gewone Differentiaalvergelijkingen, g:1/Meetkundige Problemen, h:1/Integraalrekening, i:2/Lineaire Algebra, j:0, k:0, l:0, m:1/Numerieke Wiskunde, n:0, o:2/Statistiek/Stochastiek, p:3/Mathematische Modellen/Modelleren/Systeemtheorie, q:3/Computergebruik/Programmeren, r:2/Mathematische Fysica/Mechanica; Other subjects: 3/Kaleidoscoop, 2/Krommen en Oppervlakken, 1/Oneindige Processen. Total: 28 courses.

KUN: a:4/Algebra, b:3/Analyse, c:1/Rekenen, d:1/Discrete Wiskunde, e:0, f:1/Differentiaalvergelijkingen, g:1/Meetkunde, h:0, i:0, j:1/Logica, k:0, l:1/Topologie, m:1/Numerieke Wiskunde, n:1/Besliskunde, o:1/Stochastiek, p:1/Modellenpracticum, q:2/Programmeren, r:2/Mechanica/Moderne Fysica; Other subjects: 1/Inleiding in de Wiskunde, 1/Wiskunde en Samenleving. Total: 23 courses.

good textbook has to be selected. To keep the enterprise not too complicated we suggest to start with only the 'pure' subjects, so only from the first two categories, that is only the subjects (a) until (l). (Proof checking is about proof, and proof is a primarily 'pure' mathematical activity.) This will give us twelve textbooks. Then, after selecting a proper proof checking system, these twelve books will have to be translated. We estimate that the 'size' parameter will be a suitable estimate on how many people will be needed to translate such a book: this means that the group of people needed for this enterprise will have to number about fourteen.

The advantage of this approach is that the work of 'getting the math into the system' will be separated from having to think about the order and the presentation of the mathematics. We think that a 'better' mathematical structure will result this way. Also a big advantage of this approach is that people who want to use the library will have a readable reference to it.

Here are the criteria that such a textbook has to satisfy:

- It has to be rather *thorough*. That is, it should present all of the theory, including the more basic results in the area.
- It has to be *standard*. It's far more important that the book follows accepted practice in the field than that it's mathematically elegant.
- It should *not* be *too extensive* (to keep things managable.) So it shouldn't as much be a reference to the subject as well as a text meant for teaching.

It can be argued that to digitize twelve textbooks at once might be too big for a first step: that it might be better to start small (with some specific theorem for example.) We think that (while there's some sense in this objection) the disadvantage of having to do all of undergrate mathematics at once won't be too big. It turns out that the proof of a significant theorem already uses a rather large part of the basic theory. We call this the *cone* of the theorem. (So the cone of the fundamental theorem of algebra, to name but an example, is the arithmetical, algebraic and analytical theory that is made use of in order to prove the theorem.) Our claim is that the cone of any interesting piece of mathematics will be a significant part of the basic theory that we list here.

There's a certain overlap between the subjects in our list. Of course, if that happens, the theorems from that overlap should be proven in the most basic of the subjects that 'have' it, and then be used in the others. But there is a kind of paradox here: often 'simple' theorems are generalized in later theory (e.g., analytical theorems are generalized in topology.) It's wasteful to prove both of these theorems (the basic one and the generalization.) Instead it seems natural only to prove the more advanced one and then instantiate it in the simple case. But then the dependency goes the wrong way! (By the way, this also might be a reason *not* to stay within the subjects taught to undergraduate mathematicians.)

The solution to this paradox might be to have the 'simple' theorems for the simple subjects without a reference to the more advanced versions (when using them), but to prove them – internally in the standard library – using those advanced ones. This means that then the presented theory goes from simple to advanced but the proof flow goes from advanced to simple.