Statistics on digital libraries of mathematics

Freek Wiedijk

Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. We present statistics on the standard libraries of four major proof assistants for mathematics: HOL Light, Isabelle/HOL, Coq and Mizar.

1 Introduction

1.1 Problem

The advent of digital computers has introduced a new way of doing mathematics called 'formalized mathematics'. In this style of doing mathematics one encodes the mathematics in the computer in sufficient detail that the computer can fully check the correctness according to a small number of logical rules. This style of doing mathematics is much more precise and trustable than the traditional way of first understanding the mathematics in one's head and then just writing it on a blackboard or on paper. Also it is a very pleasurable experience to write down one's mathematics in a way that *all* the details are there, knowing that there is nothing left implicit.

However, these positive aspects of formalized mathematics have to be paid for. Generally it takes much longer to turn mathematics into formalized form than it takes to just understand it, or even than to write it down in a traditional way. (A rough estimate might be that it takes about ten times as long to formalize something than it takes to write it down in meticulous traditional 'textbook style'.) One might wonder where this time is going, i.e., how much it is spent on the various aspects of formalization. For instance there are the aspects of formalizing the definitions, choosing good notation for the defined notions, then stating the appropriate formal statements to be proved, and finally writing the formal proofs themselves.

Another question that might be posed is whether there are significant differences in the time needed for these activities between the different *systems* for formalization of mathematics.

In this paper we will study these questions. We will not do this by focusing on the *activity* of formalization, but rather by studying the *results* of this activity, the libraries of formalized mathematics that have been created by the various research communities that work on this subject. These libraries have grown into quite large human 'artifacts', which – we claim – deserve study in their own right. In this paper we will do this by collecting various statistics on these libraries.

One might compare our work here to that of a biologist who just makes an inventarization of the different species that are out there in the world. In this paper we mainly just collect data.

The question that we will address here is what are the different aspects of formalization that one can find in the formalized libraries that are out there, how much of those libraries is spent on which of these aspects, and whether the different systems for formalization are more or less similar in these aspects or whether they have significant differences.

1.2 Approach

The way that we count the libraries of formalized mathematics is as follows. First we concatenate all the files for a system into one huge file. Then we tag each line of this file with the *category* of that line, and then we count the different types of lines that we find.

We will explain this procedure with a small example. Here is a very small formalization of the irrationality of the square root of two by John Harrison in the HOL Light system (taken from *The Seventeen Provers of the World* [5], a collection of formalizations of this irrationality proof in various systems):

```
(* ------------------------------------------------------------------------- *)
(* Definition of rationality (& = natural injection N->R).                   *)
(* ------------------------------------------------------------------------- *)

let rational = new_definition
  `rational(r) = ?p q. ~(q = 0) /\ abs(r) = &p / &q`;;

(* ------------------------------------------------------------------------- *)
(* The main lemma, purely in terms of natural numbers.                       *)
(* ------------------------------------------------------------------------- *)

let NSQRT_2 = prove
 (`!p q. p * p = 2 * q * q ==> q = 0`,
  MATCH_MP_TAC num_WF THEN REWRITE_TAC[RIGHT_IMP_FORALL_THM] THEN
  REPEAT STRIP_TAC THEN FIRST_ASSUM(MP_TAC o AP_TERM `EVEN`) THEN
  REWRITE_TAC[EVEN_MULT; ARITH] THEN REWRITE_TAC[EVEN_EXISTS] THEN
  DISCH_THEN(X_CHOOSE_THEN `m:num` SUBST_ALL_TAC) THEN
  FIRST_X_ASSUM(MP_TAC o SPECL [`q:num`; `m:num`]) THEN
  POP_ASSUM MP_TAC THEN CONV_TAC SOS_RULE);;

(* ------------------------------------------------------------------------- *)
(* Hence the irrationality of sqrt(2).                                       *)
(* ------------------------------------------------------------------------- *)

let SQRT_2_IRRATIONAL = prove
 (`~rational(sqrt(&2))`,
  SIMP_TAC[rational; real_abs; SQRT_POS_LE; REAL_POS; NOT_EXISTS_THM] THEN
  REPEAT GEN_TAC THEN DISCH_THEN(CONJUNCTS_THEN2 ASSUME_TAC MP_TAC) THEN
  DISCH_THEN(MP_TAC o AP_TERM `\x. x pow 2`) THEN
  ASM_SIMP_TAC[SQRT_POW_2; REAL_POS; REAL_POW_DIV; REAL_POW_2; REAL_LT_SQUARE;
               REAL_OF_NUM_EQ; REAL_EQ_RDIV_EQ] THEN
  ASM_MESON_TAC[NSQRT_2; REAL_OF_NUM_EQ; REAL_OF_NUM_MUL]);;
```

Now for each system studied in this paper, which includes the HOL Light system, we wrote a small perl script that tags each line in a formalization with its category. In our investigation we applied it to the full HOL Light library, but this is what we get when we tag just this example formalization with it:

```
C (* ------------------------------------------------------------------------- *)
C (* Definition of rationality (& = natural injection N->R).                   *)
C (* ------------------------------------------------------------------------- *)
B
D let rational = new_definition
D   `rational(r) = ?p q. ~(q = 0) /\ abs(r) = &p / &q`;;
B
C (* ------------------------------------------------------------------------- *)
C (* The main lemma, purely in terms of natural numbers.                       *)
C (* ------------------------------------------------------------------------- *)
B
T let NSQRT_2 = prove
T  (`!p q. p * p = 2 * q * q ==> q = 0`,
P   MATCH_MP_TAC num_WF THEN REWRITE_TAC[RIGHT_IMP_FORALL_THM] THEN
P   REPEAT STRIP_TAC THEN FIRST_ASSUM(MP_TAC o AP_TERM `EVEN`) THEN
P   REWRITE_TAC[EVEN_MULT; ARITH] THEN REWRITE_TAC[EVEN_EXISTS] THEN
P   DISCH_THEN(X_CHOOSE_THEN `m:num` SUBST_ALL_TAC) THEN
P   FIRST_X_ASSUM(MP_TAC o SPECL [`q:num`; `m:num`]) THEN
P   POP_ASSUM MP_TAC THEN CONV_TAC SOS_RULE);;
B
C (* ------------------------------------------------------------------------- *)
C (* Hence the irrationality of sqrt(2).                                        *)
C (* ------------------------------------------------------------------------- *)
B
T let SQRT_2_IRRATIONAL = prove
T  (`~rational(sqrt(&2))`,
P   SIMP_TAC[rational; real_abs; SQRT_POS_LE; REAL_POS; NOT_EXISTS_THM] THEN
P   REPEAT GEN_TAC THEN DISCH_THEN(CONJUNCTS_THEN2 ASSUME_TAC MP_TAC) THEN
P   DISCH_THEN(MP_TAC o AP_TERM `\x. x pow 2`) THEN
P   ASM_SIMP_TAC[SQRT_POW_2; REAL_POS; REAL_POW_DIV; REAL_POW_2; REAL_LT_SQUARE;
P                REAL_OF_NUM_EQ; REAL_EQ_RDIV_EQ] THEN
P   ASM_MESON_TAC[NSQRT_2; REAL_OF_NUM_EQ; REAL_OF_NUM_MUL]);;
B
```

The lines with a '`C`' are comment lines, the ones with a '`B`' are blank, and so on. The perl script is *ad hoc* in the sense that occasionally it will tag a line wrong. However, by inspecting its output we improved it until it was sufficiently good for our purposes. We claim that our perl scripts will tag less than 1% of the lines in a formalization with the wrong tag.

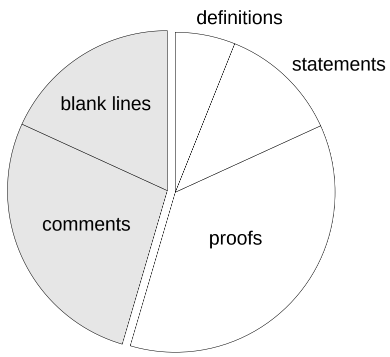
Finally we count the lines in this 'tagged' file for the various tags, and present the results in tabular format. For this small example that table then becomes:

		lines	*bytes*	
`B`	blank lines	6	65	18 %
`C`	comment lines	9	720	27 %
`D`	definitions	2	85	6 %
`T`	theorem statements	4	114	12 %
`P`	proof lines	12	746	36 %
	total	33	1,730	100 %

Of course for a very small example like this, the percentages are not very meaningful. For instance, the number of blank and comment lines is quite a bit higher than it is in a more extensive HOL Light formalization.

The percentages in this table are in terms of line counts, and not in terms of bytes. We believe that line counts is the more interesting measure. It indicates how much one can oversee behind a computer screen without scrolling. (This means that a formalization style where multiple steps are put together on a single line – a formalization style that both John Harrison and Georges Gonthier use – is superior to a more 'programming' like style in which each step gets a line of its own. Georges Gonthier convinced us in a personal communication that line counts is the better way to measure formalizations.)

Finally, we also present the table as a pie chart, as a graphical summary of the results. For these charts the different kinds of lines are grouped together into only seven categories. (In the example two of these categories are missing.) The pie chart for the example is:



Example

1.3 Related work

We are not aware of already existing research into the statistics of formalizations.

In the field of programming, counting lines of source code is one of the methods in the subject called *software metrics*. However, there generally the focus is not on the different *kinds* of source code lines and their function in the programming languages, but more on programmer productivity.

1.4 Contribution

The investigation presented here is a snapshot in time. Also there is not too much 'depth' in our results: really all we did was count. However getting the software that tagged the formalizations reasonably accurate took quite an effort, so obtaining the numbers in this paper was significant work.

The main value of the research described in this paper is showing that all systems for formalizations are quite similar despite their large differences both in foundations and in interaction styles.

The observations in this paper might be a guide for people who design systems for formalizations, by pointing out from the start which elements will need to be part of their formalization language. That way, these elements can all be designed in from the start and will not have to appear as an afterthought later.

Finally this paper can be used as a guide for people who are interested in formalization of mathematics and want to get an impression of the current state of some important libraries.

1.5 Outline

The structure of this paper is as follows. In Section 2 we present an overview of our results. Then in Section 3 we give the statistics in detail. In Section 4 we discuss the different types of lines that we distinguished, and relate them between systems. Finally, in Section 5 we draw some conclusions from our data and indicate possible future work.

2 Overview

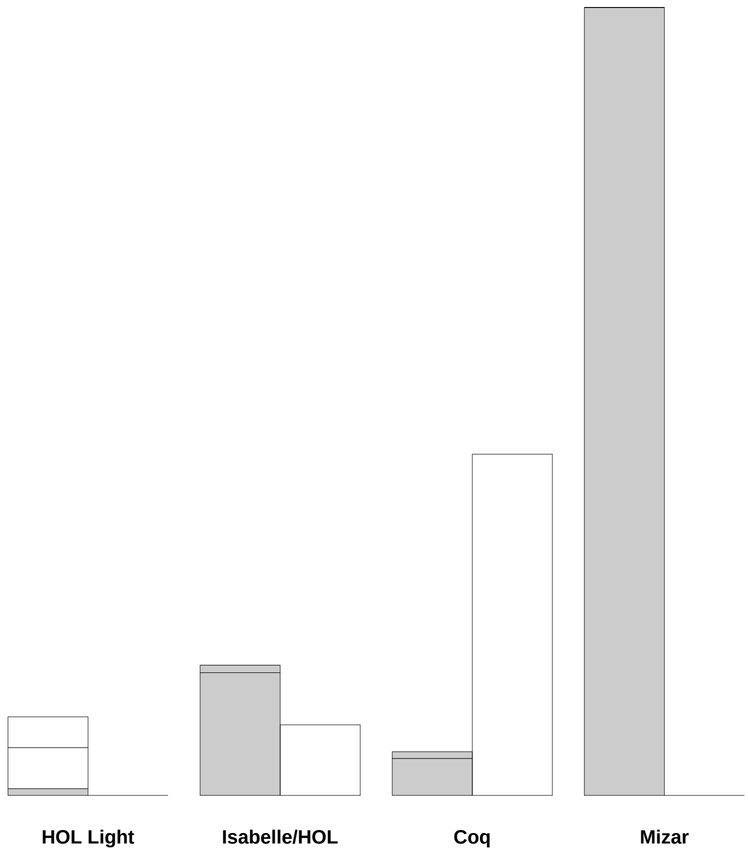
The four systems that we selected for this investigation were:

- HOL Light [1]
- Isabelle/HOL [3]
- Coq [4]
- Mizar [2]

Other systems that we considered were HOL4, ProofPower and PVS. The first two are rather similar to HOL Light, so we did expect them to get quite similar statistics. In the HOL family of systems HOL Light is the system that has been used most for formalization of mathematics. For this reason we selected HOL Light from that group, and left the other two systems out.

The PVS system is one of the most popular systems for formal methods in computer science. It has a very interesting way of dealing with partial functions (called 'predicate sub-types'), and it has strong automation. However it has not been used as much for formalization of mathematics as the other systems that we looked at. For this reason we left PVS out of our comparison as well.

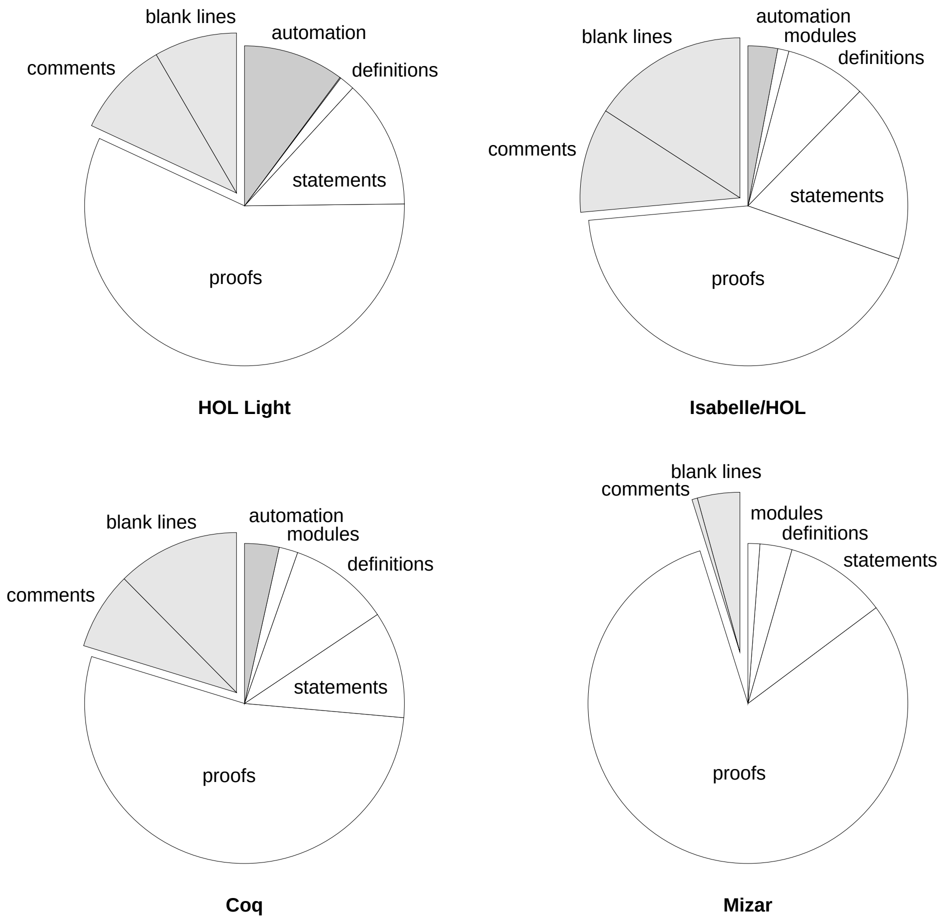
The four systems that we study here have mathematical libraries of quite different sizes. The sizes of these libraries is shown in the bar diagram on the next page. In this diagram the left bars represent the libraries that are distributed with the system. If you install the system, you will have these source files as part of the distribution.



In these bars the gray parts are the parts of the library that can be considered to be the library of the 'core' system. It is that part of the library of the system that is available without doing anything special. In the bar for the HOL Light system the rest of the library (the white part of the bar) has been divided into two subparts: the part written by John Harrison, and the part written by other people (which is mostly the formalization of the Jordan Curve Theorem by Tom Hales.) In the gray bar of the Isabelle system the small part at the top corresponds to the sub-directory of the contribution called '`HOLCF`', while in the Coq system it corresponds to the sub-directory '`contrib`'.

The right bars represent libraries that are distributed separately from the system itself. These are collections of formalizations by users of the system that have not (yet) been integrated into the standard library of the system itself. In the case of Coq this is called the Coq *contribs* (short for 'contributions'), while in the Isabelle community it is called the AFP (Archive of Formal Proofs). The Mizar system also has a library of user formalizations called MML (= Mizar Mathematical Library), but in the case of Mizar those formalizations *are* integrated into the standard library of the system.

We now show a summary of the statistics from this paper in the shape of four pie charts:



For most people the bar chart on the previous page and these pie charts will be the most interesting part of this paper.

In the HOL Light pie chart there is a tiny sliver between 'automation' and 'definitions' for the very few lines related to 'modules', but it was too narrow to be labeled. In the Mizar pie chart the 'registration' lines have been included in the 'statements' part, although we gave them category '`H`' (which in Section 4 is in the sub-section about automation; we will discuss this there in more detail.)

3 Line counts

We now present the detailed statistics of the four systems, and discuss which files were counted and which were not.

3.1 HOL Light

The statistics in this paper are on version 2.20 of the HOL Light system.

HOL Light input files have suffix '`.ml`'. These files both contain the implementation of the system as well as the mathematical library, all mixed together. We divided the files that were primarily implementing the system from files that were primarily proving the library in the following way:

The basic HOL Light system consists of a file called `make.ml`, which loads the main file `hol.ml`, which then loads 44 more `.ml` files. (Apart from these 46 files there are in the top level directory 14 more `.ml` files with names of the form '`pa_j_`....`ml`' related to the input processing of the `ocaml` system that reads the HOL Light files.)

Now the `hol.ml` file is divided into sections. One of these sections has the header 'Mathematical theories and additional proof tools.' We decided that the 19 files in that section were the 'mathematical library' while the 25 files in the other six sections contained the 'implementation of the system'.

Apart from these 19 files we also counted the 'auxiliary library' consisting of 169 `.ml` files in 11 sub-directories. (There were 11 more `.ml` files in another sub-directory named `Proofrecording`. However that is an alternative implementation of the core system, and therefore was left out of this investigation.)

Altogether the number of files counted were:

19 files	*from* `*.ml`
169 files	`*/*.ml`
188 files	

And the statistics about these files were:

		lines	*bytes*	
`B`	blank lines	16,438	21,371	8.4 %
`C`	comments	19,044	864,913	9.7 %
`S`	imports	182	5,459	0.1 %
`D`	definitions	2,547	107,835	1.3 %
`N`	interfaces	486	19,525	0.2 %
`X`	automation: program code	19,979	833,040	10.2 %
`T`	theorem statements	25,493	1,073,208	13.0 %
`P`	proof lines	112,088	4,356,332	57.1 %
	total	196,257	7,281,683	100.0 %

3.2 Isabelle/HOL

The statistics in this paper are on the Isabelle2007 version of the Isabelle/HOL system.

Isabelle has two kinds of files, with suffixes '`.ML`' and '`.thy`'. We decided that the `.ML` files primarily contained the implementation of the system, while the

`.thy` files primarily contained the mathematical library. Now the Isabelle system can be used with different *logics*. The HOL logic is the dominant logic that is used by almost all Isabelle users. For this reason we counted the `.thy` files inside the `HOL` directory (together with `HOLCF` directory, which is closely related). However, we left out the `HOL/Import` sub-directory as it does not contain mathematics but is about importing theories from other systems.

The number of files counted were (here `**` stands for zero or more sub-directory levels in between):

649 files	*most of*	`src/HOL/**/*.thy`
88 files		`src/HOLCF/**/*.thy`
737 files		

And the statistics about these files were:

		lines	*bytes*	
`B`	blank lines	51,610	80,304	15.9 %
`C`	source comments	18,941	829,132	5.8 %
`E`	document markup	15,488	713,503	4.8 %
`S`	imports & sectioning	2,838	43,584	0.9 %
`L`	locales	1,394	58,862	0.4 %
`D`	definitions	23,386	1,165,813	7.2 %
`N`	notation	2,736	129,089	0.8 %
`H`	automation: directives	4,022	191,544	1.2 %
`X`	automation: program code	5,714	225,786	1.8 %
`T`	theorem statements	58,659	3,136,976	18.0 %
`P`	proof lines	140,596	5,024,717	43.2 %
	total	325,384	11,599,310	100.0 %

3.3 Coq

The statistics in this paper are about version 8.1 of the Coq system.

Coq files have the suffix '`.v`'. (The implementation of the system is in files with suffixes '`.mli`' and '`.ml`', but unlike HOL Light and Isabelle these are in a different part of the distribution, and are not mixed together with the mathematical library.)

The main library is in the sub-directory `theories`. There is a supplementary library in the sub-directory `contrib`, which mostly contains the supporting theory for several automated proof procedures. (In this second directory the `.v` files and the `.mli` and `.ml` files *are* together. However we also only looked at the `.v` files there.)

Altogether the number of files counted were:

252 files	`theories/*/*.v`
57 files	`contrib/*/*.v`
309 files	

And the statistics about these files were:

		lines	*bytes*	
`B`	blank lines	13,531	22,456	12.4 %
`C`	non-coqdoc comments	5,661	300,850	5.2 %
`E`	coqdoc comments	2,910	137,401	2.7 %
`S`	imports & sectioning	2,073	49,377	1.9 %
`L`	context	1,329	50,686	1.2 %
`D`	definitions	8,778	308,858	8.0 %
`N`	notation	1,047	40,293	1.0 %
`H`	automation: directives	1,157	45,680	1.1 %
`X`	automation: program code	2,648	94,556	2.4 %
`T`	theorem statements	11,781	541,836	10.8 %
`P`	proof lines	58,157	1,981,655	53.3 %
	total	109,072	3,573,648	100.0 %

3.4 Mizar

The statistics in this paper are on version 7.8.05 of the Mizar system, which is distributed with version 4.87.985 of the MML mathematical library.

Mizar files have the suffix '`.miz`'. (There also are files with suffix '`.abs`' that are 'abstracts' to the formalizations, but they are derived from the first kind of files and do not contain any independent information.) As the version number of the MML library already shows, there are 985 `.miz` files distributed with the system.

Therefore the number of files counted were:

985 files `mml/*.miz`

And the statistics about these files were:

		lines	*bytes*	
`B`	blank lines	84,609	87,744	4.3 %
`C`	comments	10,857	488,821	0.6 %
`S`	environments, cancellations	23,655	1,118,819	1.2 %
`L`	reservations	5,396	194,693	0.3 %
`D`	definitions	56,738	1,847,161	2.9 %
`N`	notation	1,634	45,598	0.1 %
`H`	registrations	26,016	749,427	1.3 %
`T`	theorem statements	177,829	6,625,141	9.0 %
`P`	proof lines	1,582,831	61,096,949	80.4 %
	total	1,969,575	72,254,353	100.0 %

4 Categories of lines in a formalization

In the previous section we tagged lines in different systems that had similar functions with the same letter. Here we identify how these letters should be interpreted.

For most of the categories we list the main keywords that are associated with the lines of that category. For a user of the system this makes it quite clear how we divided the lines among the categories. (In Mizar there was the clearest bijection between keywords starting a part of a formalization and categories in our statistics. In the other systems the correspondence was a bit less obvious.)

4.1 Non-content lines

B – blank lines. Lines tagged '`B`' are blank lines. These amount to a surprising large part of the total line count of a formalization. In the byte counts of this category we also included the white space at the end of other kinds of lines. Also sometimes some care had to be taken with files that did not end in a newline character. (For such files one newline byte was added.)

C – comments. The following table shows the comment styles found in the four systems:

HOL Light:	`(*` *comment* `*)`
Isabelle/HOL:	`(*` *comment* `*)`
Coq:	`(*` *comment* `*)`
Mizar:	`::` *comment*

E – documentation Isabelle and Coq generate documentation for the formalizations by having text inside special comments. In Isabelle these comments come in two styles, and are always prefixed with a keyword or a double dash. At first we included some of these lines in the sectioning category below, but Makarius Wenzel convinced us in private communication that they really belong in this category.

Isabelle/HOL:	`header section subsection subsubsection text txt --`
	`{*` *text* `*}` `"`*text*`"`
Coq:	`(**` *text* `*)`

4.2 Modules

Imports, sectioning and modules seem closely related, but there is a gray area with the notion of definitions. For instance in Isabelle a 'locale' might be considered to group related definitions together, but it also might be considered to consist of definitions. (We chose the second interpretation.) Similarly Coq modules seem rather close to Coq structures. (We chose to consider the first to be about modularization and the second to be a data-type definition.)

S – imports and sectioning. We did not distinguish between lines that group parts of a formalization together into a section or module, and lines that open or import these sections or modules.

The main keywords for this category of lines in the four systems were:

HOL Light: `needs loadt`
Isabelle/HOL: `theory imports begin use uses`
Coq: `Require Section Module Import`
Mizar: `environ begin canceled`

One could also consider Isabelle's `use` and `uses` to belong to the automation category below, but we decided to consider them to be import lines.

4.3 Definitions

L – contexts. The '`L`' lines build 'contexts' in which a definition can be made. We considered these lines to be part of those definitions. In the Isabelle system these contexts are named entities. In Coq they just are implicit through the position in the section or module. In Mizar we used this letter for lines that introduce variable conventions.

Isabelle/HOL: `class locale context`
`instance interpret interpretation`
Coq: `Variable Variables Hypothesis Parameter Axiom`
Mizar: `reserve`

D – definitions. The systems all have numerous constructions for defining functions, predicates and types. Here are the main keywords for these constructions:

HOL Light: `new_definition new_recursive_definition define`
`new_inductive_definition new_specification`
`new_type_definition`
Isabelle/HOL: `abbreviation axclass coinductive constdefs consts`
`datatype definition defs fun function inductive`
`inductive_set nominal_datatype nominal_inductive`
`nominal_primrec primrec recdef record specification`
`typedecl typedef types`
Coq: `Definition Fixpoint Inductive CoFixpoint CoInductive`
`Record Function`
Mizar: `definition`

N – notation. These are the lines that direct the parser and pretty-printer of the system. These lines do not define the notions themselves, but introduce the syntax for the defined notions.

HOL Light: `parse_as_infix unparse_as_infix parse_as_binder make_overloadable overload_interface override_interface reduce_interface prioritize_num prioritize_real`
Isabelle/HOL: `syntax translations notation nonterminals parse_translation print_translation`
Coq: `Infix Notation` ‘`Reserved Notation`’ ‘`Tactic Notation`’ `Coercion` ‘`Implicit Arguments`’ ‘`Set Implicit Arguments`’ ‘`Unset Implicit Arguments`’ ‘`Set Strict Implicit`’ ‘`Unset Strict Implicit`’ ‘`Open Scope`’ ‘`Open Local Scope`’
Mizar: `notation`

4.4 Automation

The automation of a system has two kinds of lines. First there are the lines that set parameters for the automated decision procedures and proof search procedures. Second there are the implementations of these automated procedures.

Most of the automation is implemented outside of the formalizations and is not counted here, but procedures that are specific to the subject are often implemented inside the formalization.

H – automation: directives. The automation ‘directives’ often are mixed with statements. For instance, in Isabelle theorem statements can be annotated with ‘`[simp]`’. This really is an automation directive, but it does not have a line of its own, so it will not be reflected in the statistics for this category of lines. Similarly, the Mizar ‘registrations’ (which direct the automation of the Mizar type system) also can be read as statements. For this reason in the Mizar pie chart on page 7 this category was included in the group about statements, and not in a group about automation.

Isabelle/HOL: `declare lemmas theorems`
Coq: `Hint Add Opaque Transparent Scheme`
Mizar: `registration`

X – automation: program code. These lines are implementations of proof procedures. In the HOL Light system really *all* lines are in some sense in this category, as a HOL Light formalization really just is an OCaml program. Therefore in the case of HOL Light the lines of this category are what remains when the other categories are removed.

The Mizar system does not have this kind of line as Mizar does not support user level proof automation.

HOL Light: `let`
Isabelle/HOL: `ML ML_setup declaration method_setup oracle setup simproc_setup`
Coq: `Ltac`

4.5 Theorems

A formalization mainly consists of a long chain of 'lemmas'. These lemmas generally consist of a label, a statement and a proof.

T – theorem statements. In this category are the lines which state the theorem and give its label.

The Mizar system actually distinguishes between two kinds of statements: theorems and schemes. The first category are the first order statements, while the second category are the higher order statements. Here we do not distinguish between these two categories.

HOL Light:	`prove prove_by_refinement`
Isabelle/HOL:	`lemma theorem inductive_cases axioms axiomatization corollary subclass termination`
Coq:	`Lemma Theorem Goal`
Mizar:	`theorem scheme`

P – proofs. Finally there are the lines of the formalized proofs. As is apparent in the pie charts in Section 2 these lines amount to about half of the formalizations. These lines contain many different constructions all with their own keywords. Here we just give the keywords that bracket the proofs.

Isabelle/HOL:	`apply by proof qed done oops sorry`
Coq:	`Proof Qed Save Defined`
Mizar:	`proof end`

5 Conclusions

5.1 Discussion

The three main conclusions of this study for us are:

– The four systems are quite similar. Despite a large difference in foundations (the HOL logic, the Calculus of Inductive Constructions and Tarski-Grothendieck set theory are all quite different) and in interaction style (talking to an OCaml interpreter, interacting with a tactic prover in a Proof General style interface and using a compiler-like batch checker), the actual formalizations all share the same elements.
– The HOL Light system has the smallest definition segment in its pie chart. This seems to suggest that it is the most reliable. Andrzej Trybulec taught me in private communication that a definition is like a *debt*, because you do not know whether what you are defining corresponds to the informal notion in your head. You gain confidence in this by proving theorems about the notion later. That way you pay the debt back, and gain trust in that your

formalization actually means what you think it means. In this sense the HOL Light system is the most trustable of the four.
Another interpretation, proposed by John Harrison in a private communication, is that the low percentage of the definitions does not so much reflect the quality of the formalization but rather the fact that the HOL Light library primarily contains pure mathematics. This seems to be collaborated by the observation that the percentage of the definitions in John Harrison's verification work at Intel, also using the HOL Light system, is 3.7% instead of 1.3%.
– The Mizar system has the largest proof segment in its pie chart. This suggests that its proof language might be less efficient. (It is very natural and pleasant to use, though.) This might be related to Mizar's declarative proof style, or the fact that the Mizar system does not have much automation.

5.2 Future work

It might be interesting to delve into the 'fine structure' of the largest segment in the pie charts, the proof lines. However, it probably is hard to systematically distinguish different kinds of proof steps on a line by line basis. An interesting question about the proof lines might be how many are straight-forward 'manual' reasoning steps, and how many invoke strong automated proof procedures.

Acknowledgments. Thanks to John Harrison, Henk Barendregt and Makarius Wenzel for helpful comments. Special thanks to John Harrison for sending me statistics on his Intel verification work. Special thanks to Makarius Wenzel for sending me a list of categorized Isabelle keywords.

References

1. J.R. Harrison. *The HOL Light manual (1.1)*, 2000. ⟨`http://www.cl.cam.ac.uk/users/jrh/hol-light/manual-1.1.ps.gz`⟩.
2. Michał Muzalewski. *An Outline of PC Mizar*. Fondation Philippe le Hodey, Brussels, 1993. ⟨`http://www.cs.ru.nl/~freek/mizar/mizarmanual.ps.gz`⟩.
3. T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002. ⟨`http://www.cl.cam.ac.uk/Research/HVG/Isabelle/dist/Isabelle2004/doc/tutorial.pdf`⟩.
4. The Coq Development Team. *The Coq Proof Assistant Reference Manual*, 2006. ⟨`http://pauillac.inria.fr/coq/doc/main.html`⟩.
5. Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *LNCS*. Springer, 2006. With a foreword by Dana S. Scott.