NICTA

Large-scale proof and libraries in Isabelle/HOL

Gerwin Klein

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members
ANU THE AUSTRALIAN NATIONAL UNIVERSITY
UNSW THE UNIVERSITY OF NEW SOUTH WALES
NSW First for Business
Department of State and Regional Development
Victoria The Place To Be
THE UNIVERSITY OF MELBOURNE
The University of Sydney
Queensland Government
Griffith UNIVERSITY
QUT Queensland University of Technology
The University Of Queensland AUSTRALIA
NICTA Partners

L4Verified

L4.Verified

l microkernel

8,700 lines of C

0 bugs*

qed

*conditions apply

Windows

An exception 06 has occured at 0028:C11B3ADC in VxD DiskTSD(03) +
00001660. This was called from 0028:C11B40C8 in VxD voltrack(04) +
00000000. It may be possible to continue normally.

* Press any key to attempt to continue.
* Press CTRL+ALT+RESET to restart your computer. You will
  lose any unsaved information in all applications.

Press any key to continue

Windows Vista™

Stunning. Breakthrough. Entertaining.

HP TouchSmart PC and Microsoft Windows Vista deliver you a PC experience designed to fit wherever life happens.

Innovative solutions brought to you by: Microsoft hp BEST BUY


the Bay
the Bay

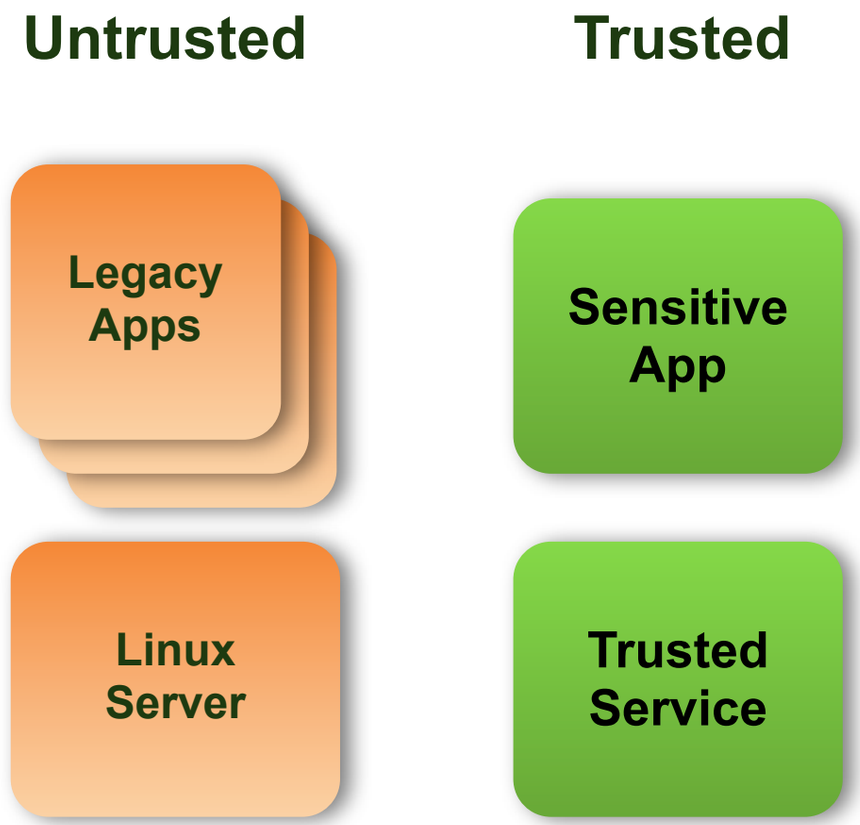
The Problem

Small Kernels

NICTA

Small trustworthy foundation

- hypervisor, microkernel, nano-kernel, virtual machine, separation kernel, exokernel ...
- High assurance components in presence of other components



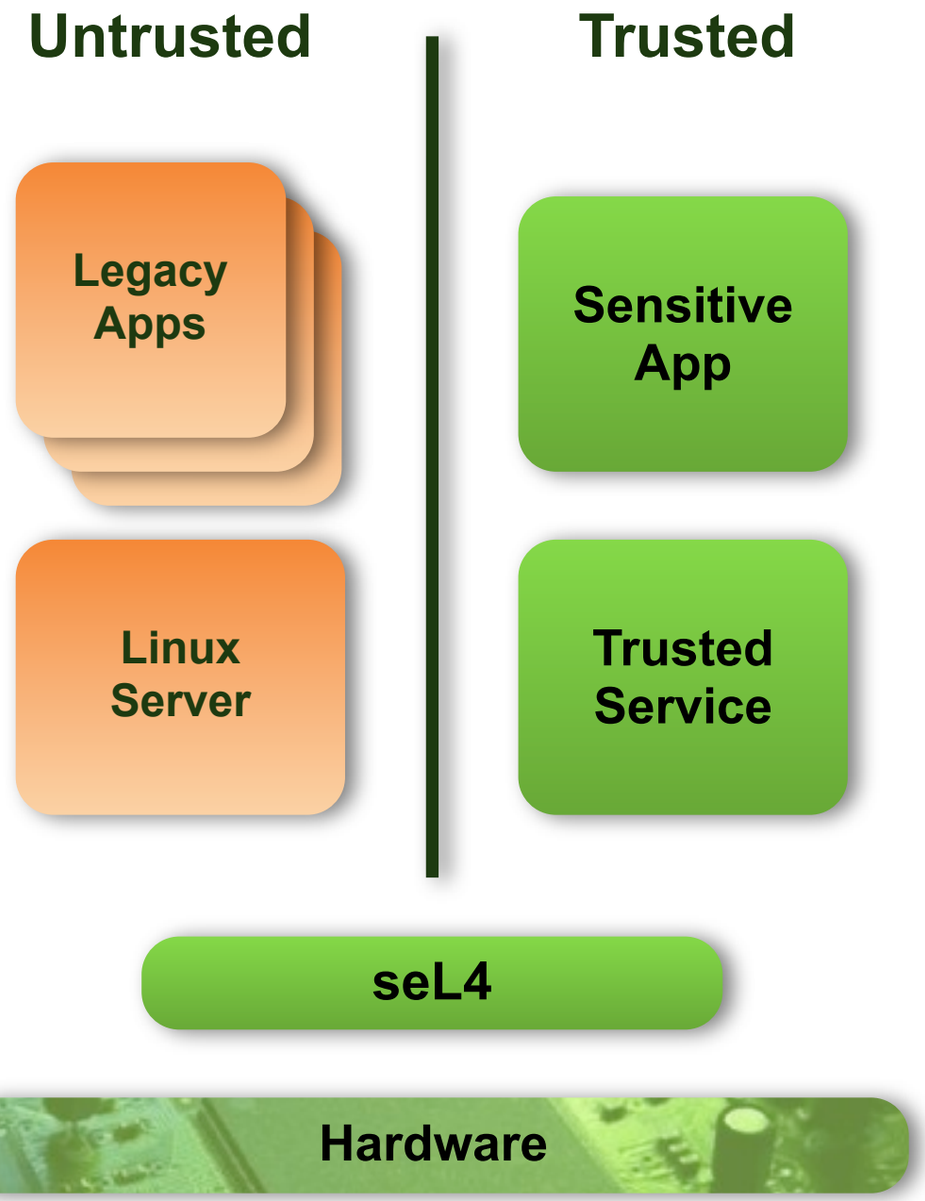
seL4 API:

- IPC
- Threads
- VM
- IRQ
- Capabilities

Small Kernels

Small trustworthy foundation

- hypervisor, microkernel, nano-kernel, virtual machine, separation kernel, exokernel ...
- High assurance components in presence of other components



seL4 API:

- IPC
- Threads
- VM
- IRQ
- Capabilities

The Proof

The Proof

Functional Correctness

Specification

Proof

Code

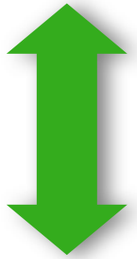
Functional Correctness

What

Proof

Specification

```
definition
  schedule :: unit s_monad where
  schedule ≡ do
    threads ← allActiveTCBs;
    thread ← select threads;
    switch_to_thread thread
  od
  OR switch_to_idle_thread
```



Code

Functional Correctness

NICTA

What

Proof

How

Specification

```
definition
  schedule :: unit s_monad where
  schedule ≡ do
    threads ← allActiveTCBs;
    thread ← select threads;
    switch_to_thread thread
  od
  OR switch_to_idle_thread
```

Implementation

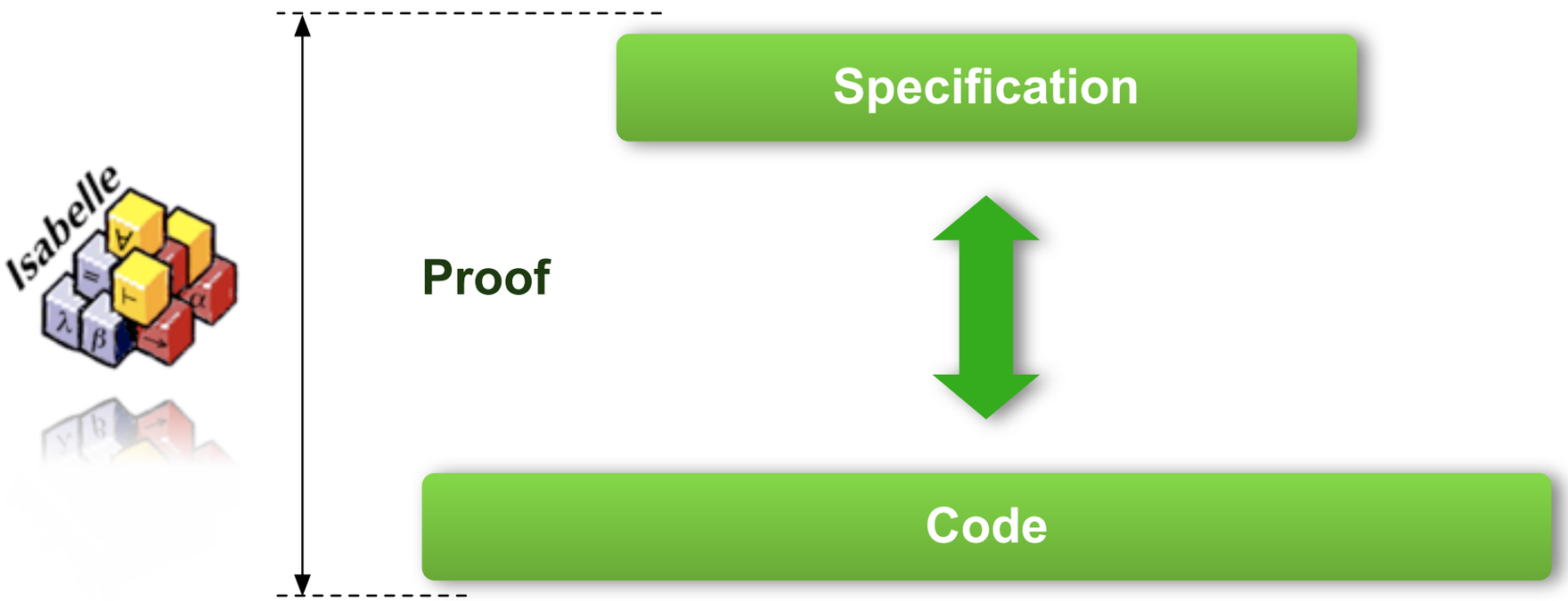
```
void
schedule(void) {
    switch ((word_t)ksSchedulerAction) {
        case (word_t)SchedulerAction_ResumeCurrentThread:
            break;

        case (word_t)SchedulerAction_ChooseNewThread:
            chooseThread();
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;

        default: /* SwitchToThread */
            switchToThread(ksSchedulerAction);
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;
    }
}

void
chooseThread(void) {
    prio_t prio;
    tcb_t *thread, *next;
```

*conditions apply



*conditions apply

NICTA

Expectation

Specification

Proof

Code

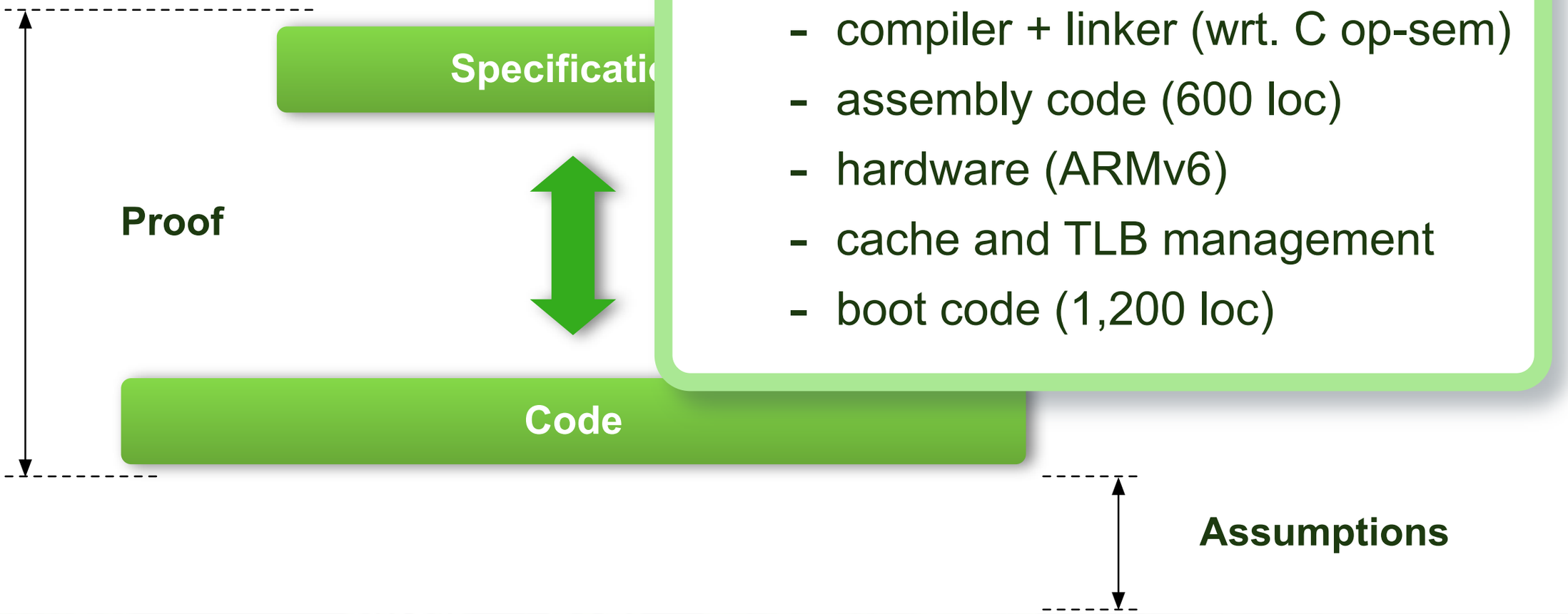
Assumptions

*conditions apply

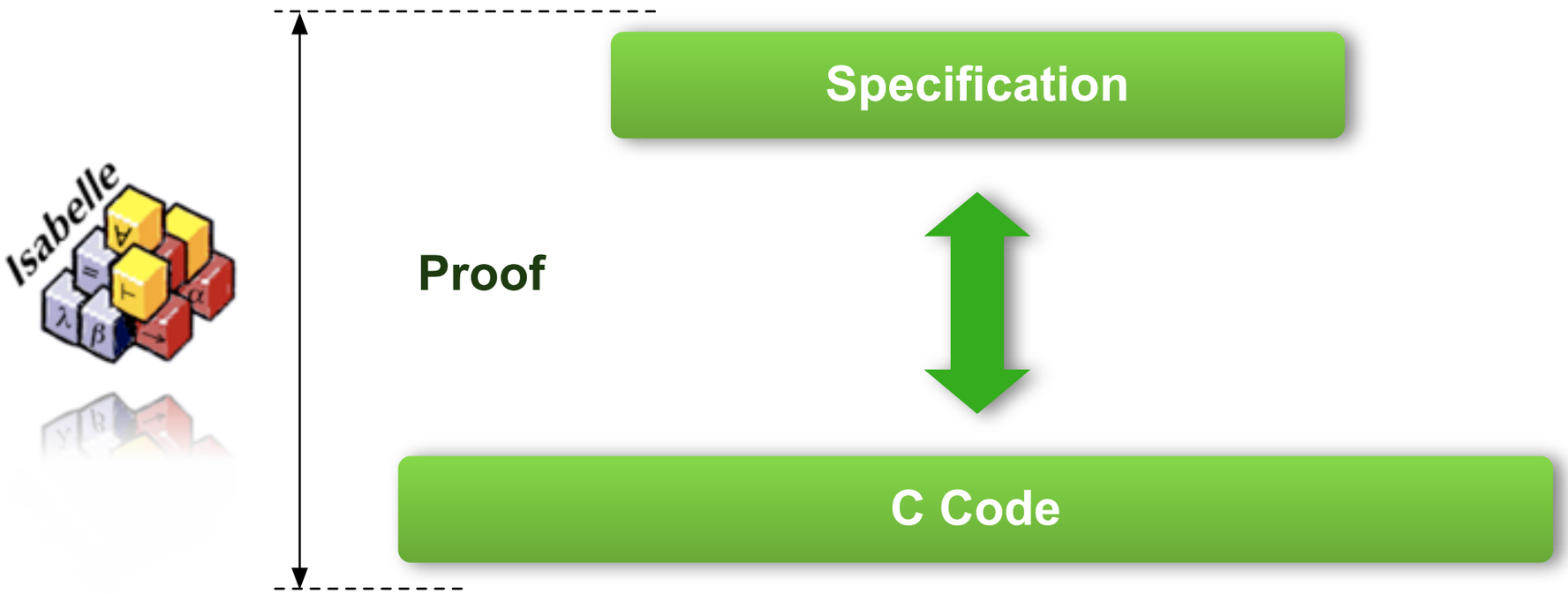
NICTA

Assume correct:

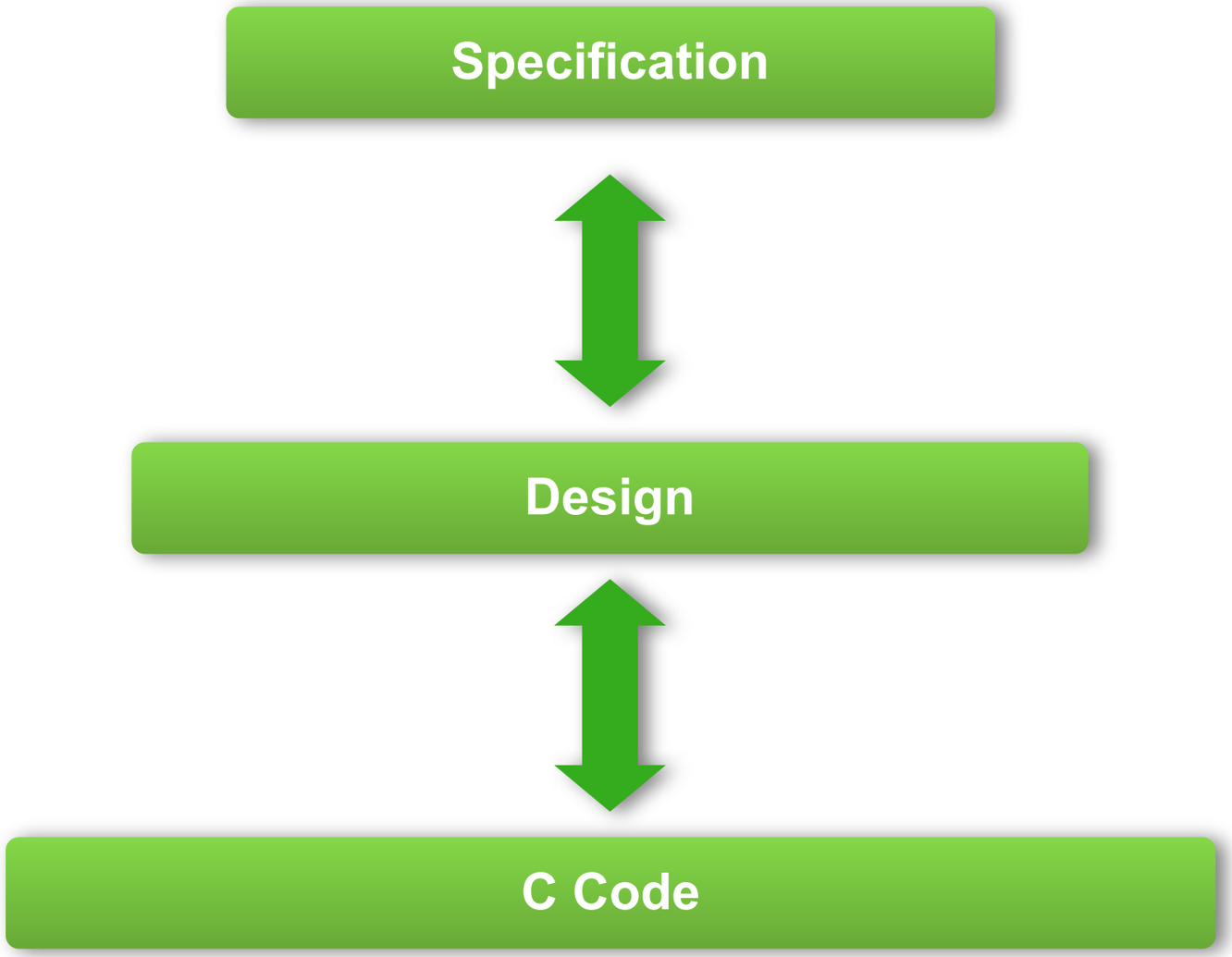
- compiler + linker (wrt. C op-sem)
- assembly code (600 loc)
- hardware (ARMv6)
- cache and TLB management
- boot code (1,200 loc)



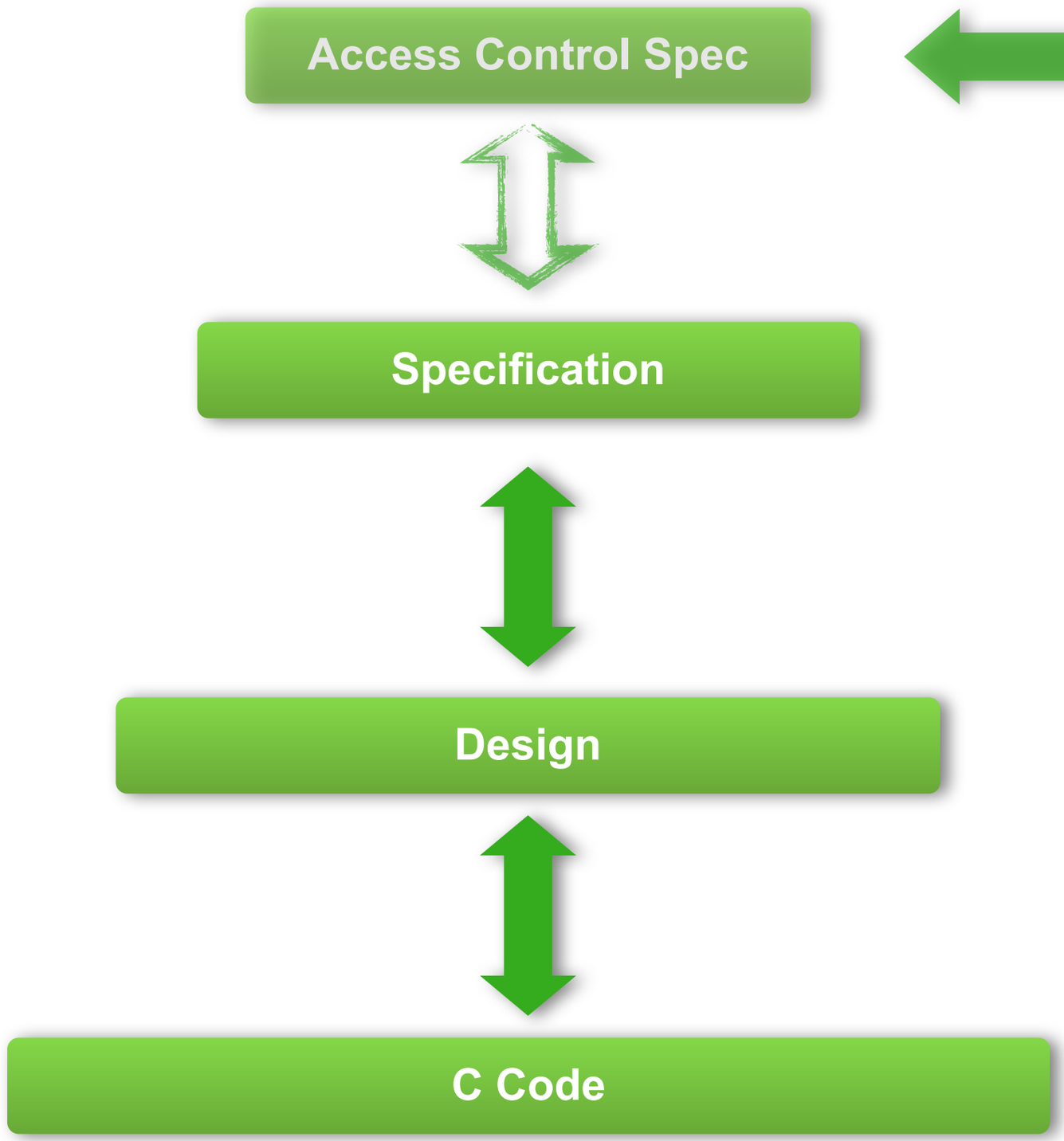
Proof Architecture



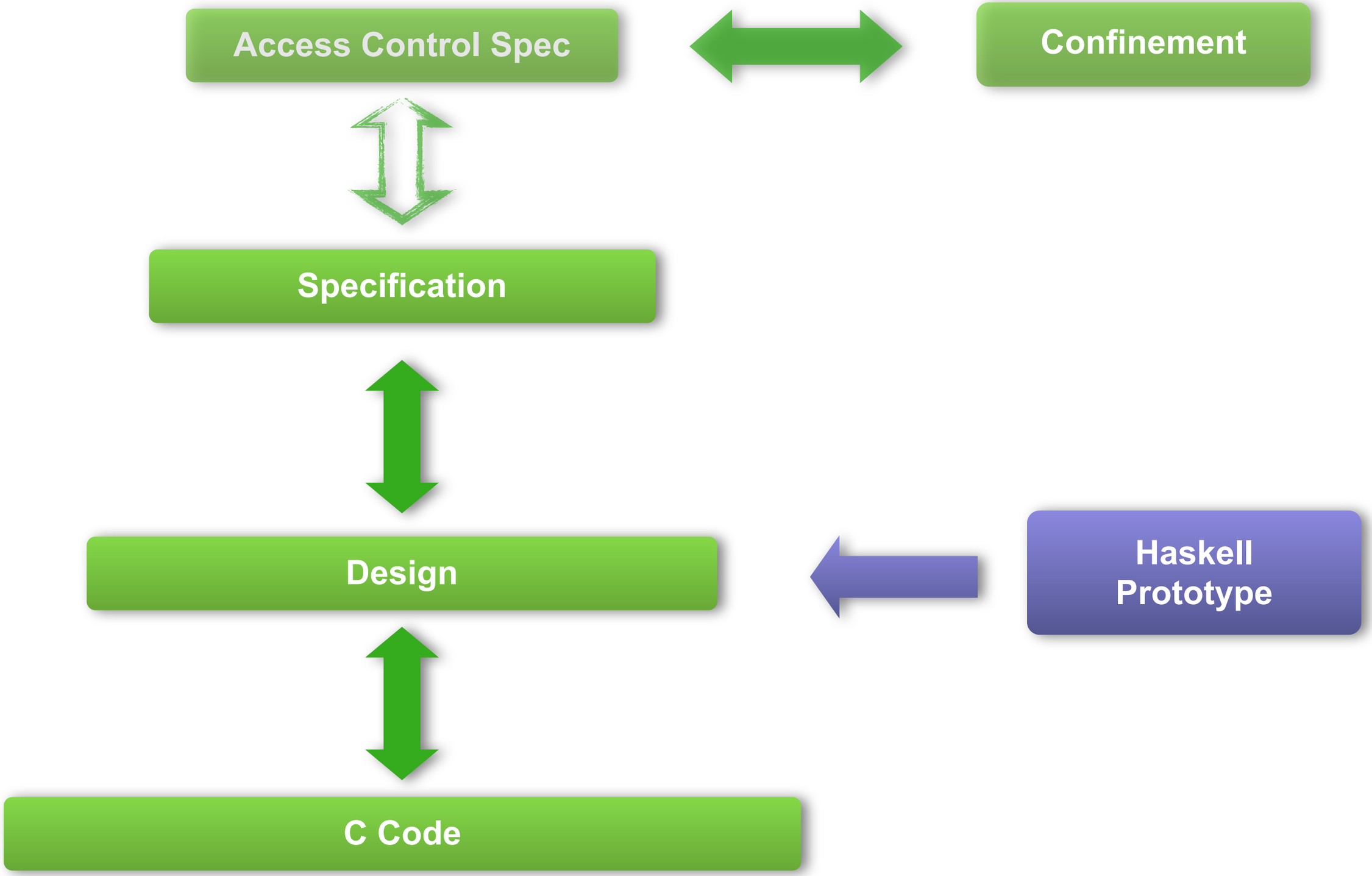
Proof Architecture



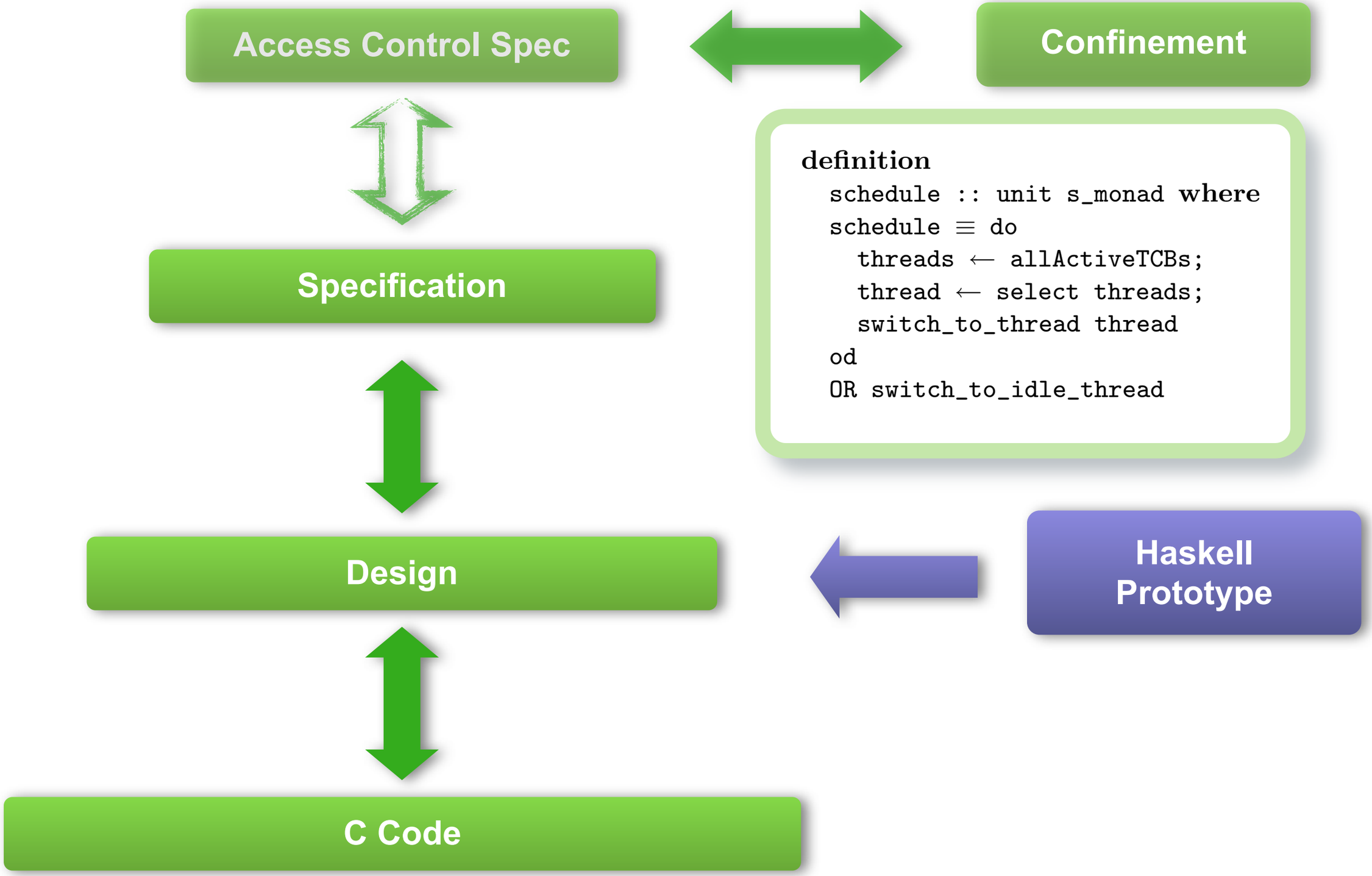
Proof Architecture



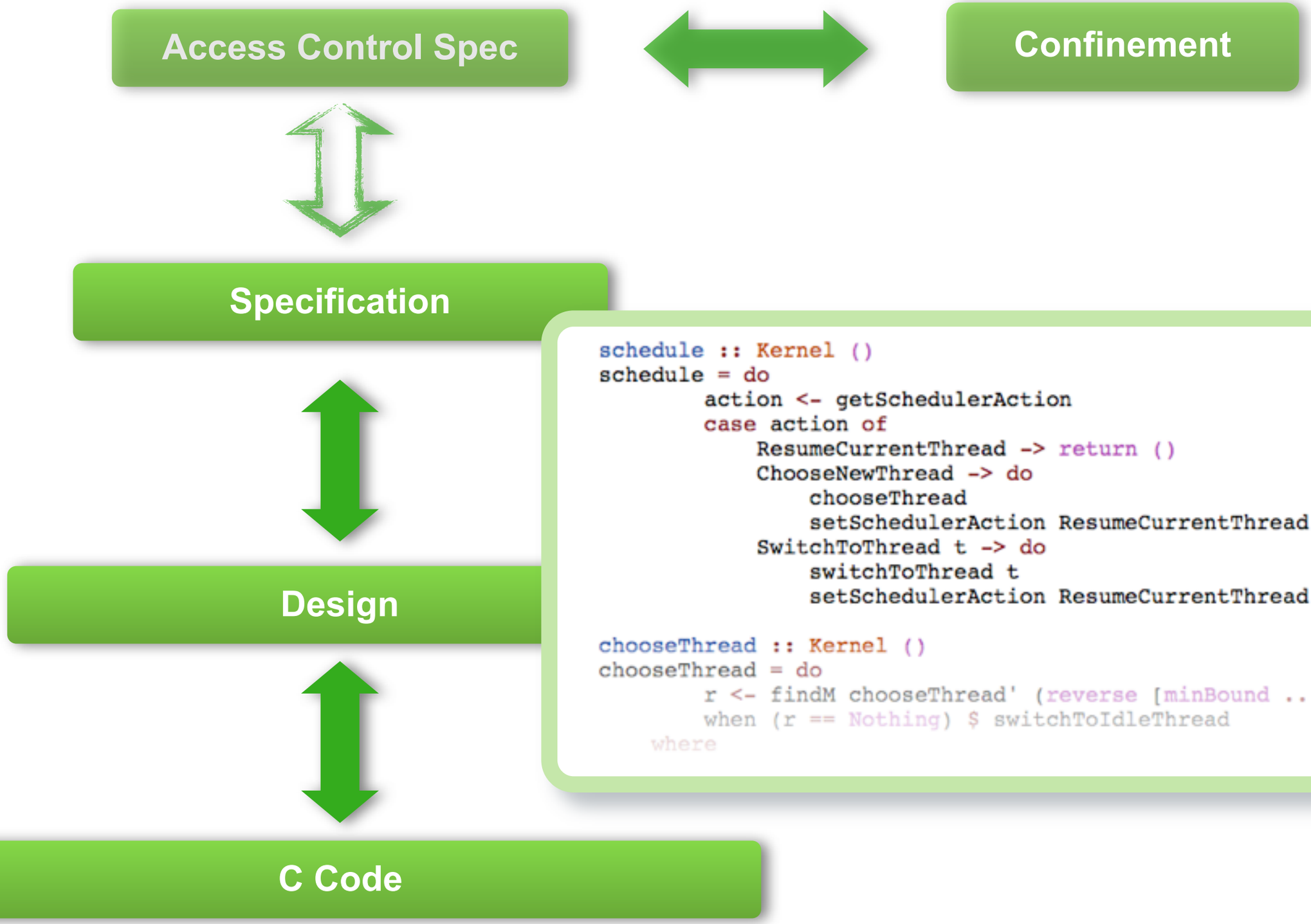
Proof Architecture



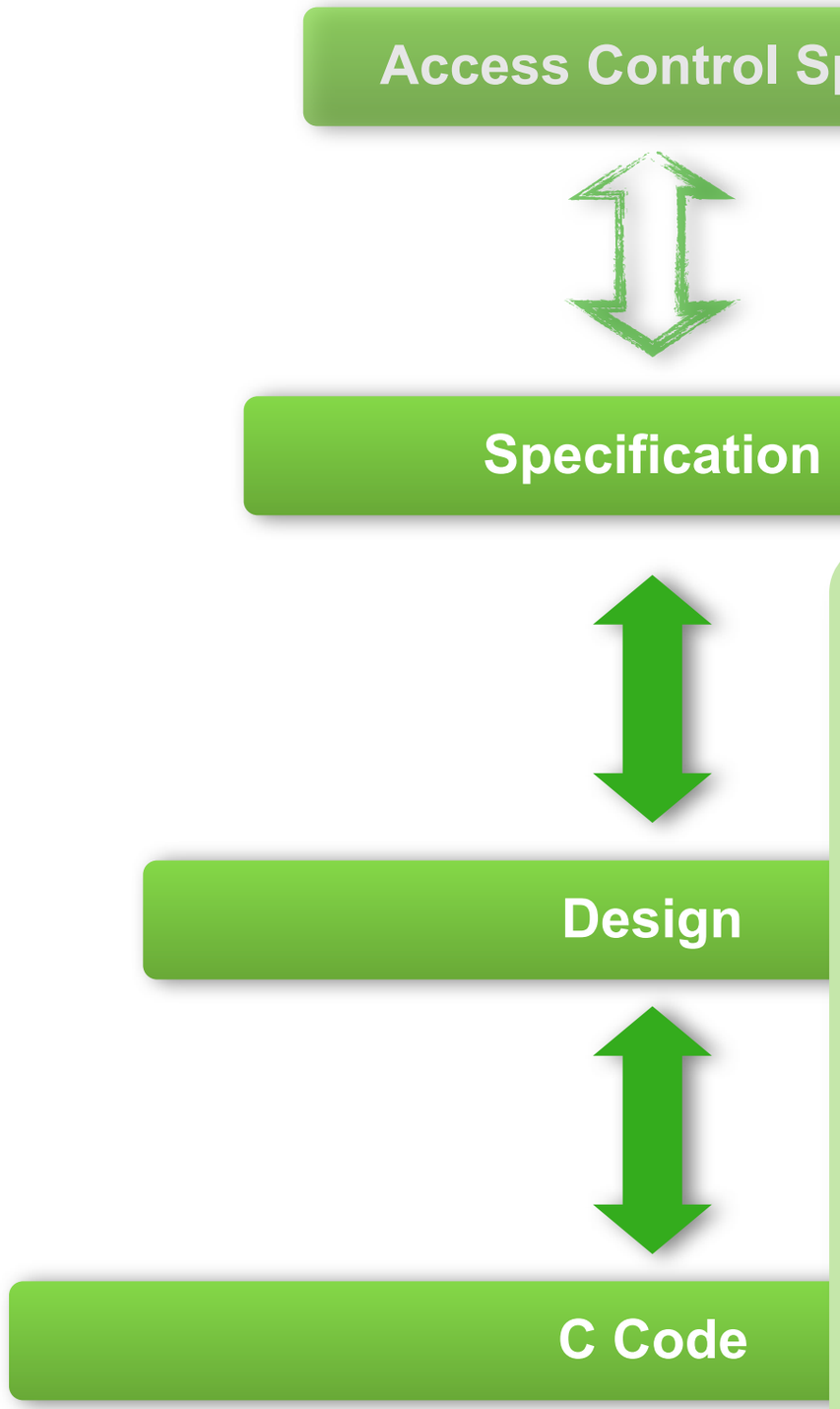
Proof Architecture



Proof Architecture



Proof Architecture

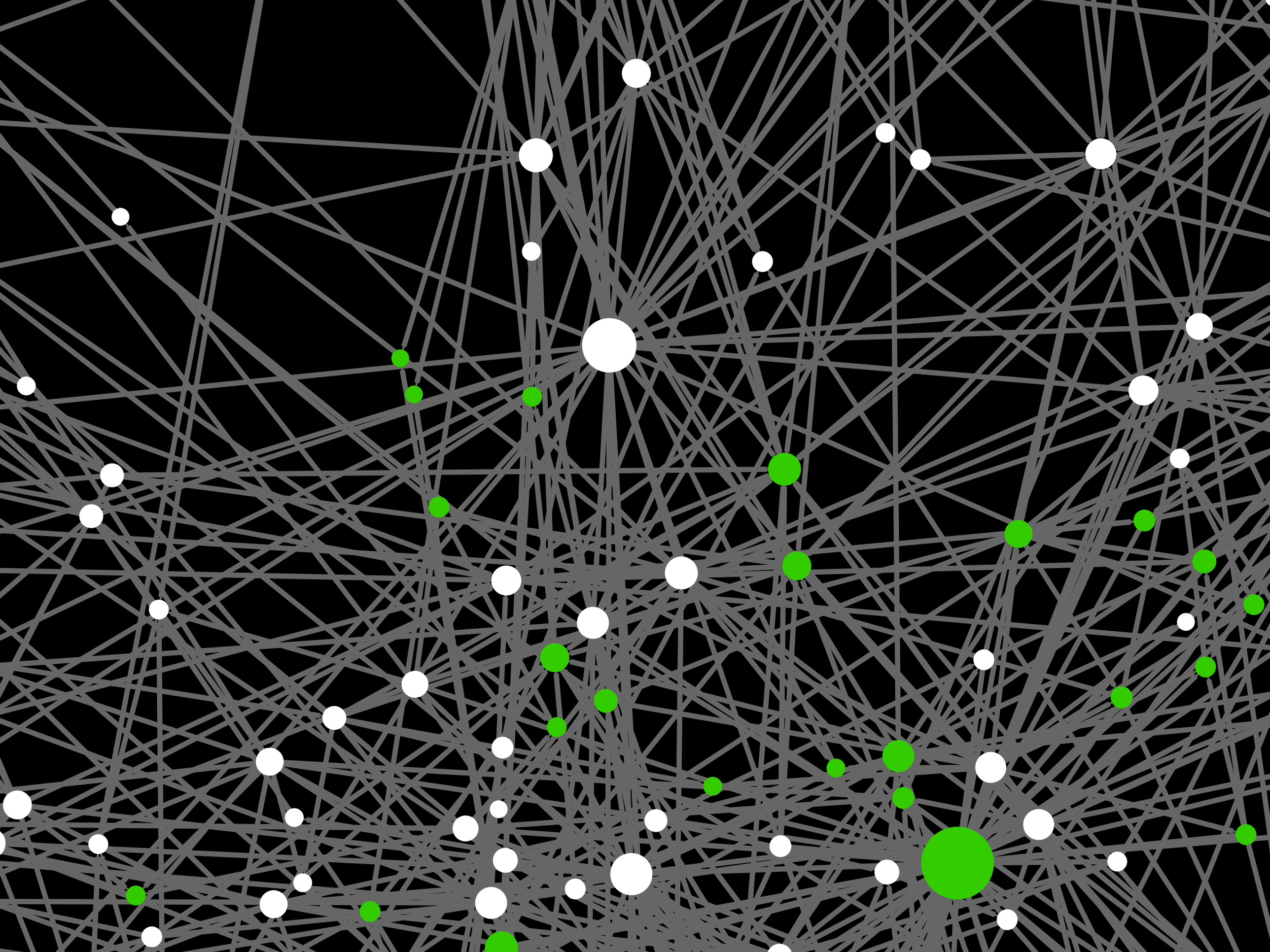



```
void
schedule(void) {
    switch ((word_t)ksSchedulerAction) {
        case (word_t)SchedulerAction_ResumeCurrentThread:
            break;

        case (word_t)SchedulerAction_ChooseNewThread:
            chooseThread();
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;

        default: /* SwitchToThread */
            switchToThread(ksSchedulerAction);
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;
    }
}

void
chooseThread(void) {
    prio_t prio;
    tcb_t *thread, *next;
```




seL4

Did you find any Bugs?

NICTA

Bugs found

- in C: 160
- in design: ~150
- in spec: ~150

460 bugs

```
void
schedule(void) {
    switch ((word_t)ksSchedulerAction) {
        case (word_t)SchedulerAction_ResumeCurrentThread:
            break;

        case (word_t)SchedulerAction_ChooseNewThread:
            chooseThread();
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;

        d

    }
}

void
chooseThr
    prio_
    tcb_t

    for(p
        f

                if(!isRunnable(thread)) {
                    next = thread->tcbSchedNext;
                    tcbSchedDequeue(thread);
                }
                else {
                    switchToThread(thread);
                    return;
                }
            }
        }
```

ead;

Effort

Haskell design	2	py
First C impl.	2	weeks
Debugging/Testing	2	months
Kernel verification	12	py
Formal frameworks	10	py
Total	25	py

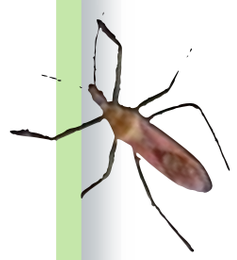
Did you find any Bugs?

NICTA

Bugs found

- in C: 160
- in design: ~150
- in spec: ~150

460 bugs



```
void
schedule(void) {
    switch ((word_t)ksSchedulerAction) {
        case (word_t)SchedulerAction_ResumeCurrentThread:
            break;

        case (word_t)SchedulerAction_ChooseNewThread:
            chooseThread();
            ksSchedulerAction = SchedulerAction_ResumeCurrentThread;
            break;
    }
}

void
chooseThr
    prio_
    tcb_t

    for(p
        f
                if(!isRunnable(thread)) {
                    next = thread->tcbSchedNext;
                    tcbSchedDequeue(thread);
                }
                else {
                    switchToThread(thread);
                    return;
                }
            }
        }
```

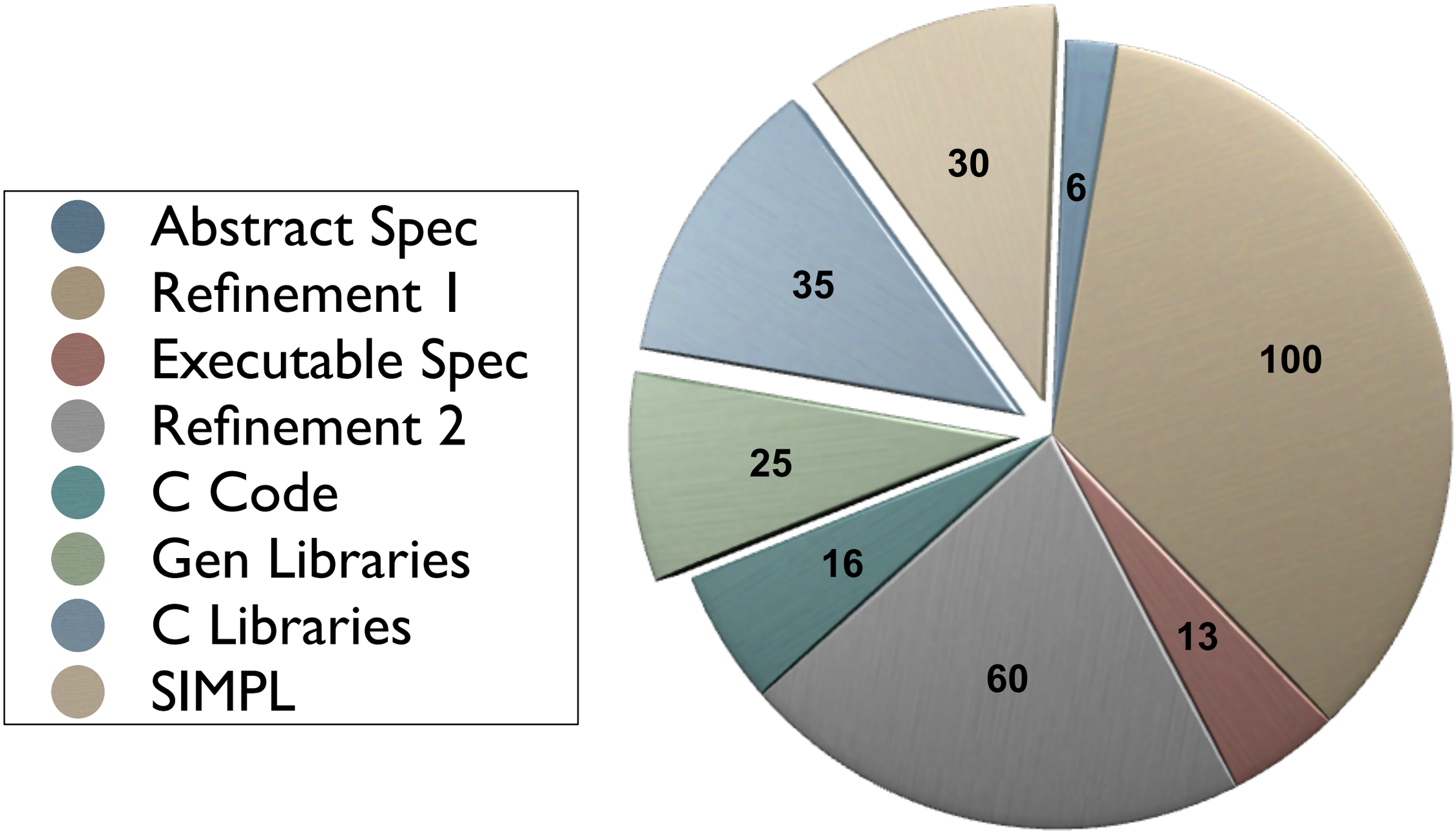
Effort

Haskell design	2	py
First C impl.	2	weeks
Debugging/Testing	2	months
Kernel verification	12	py
Formal frameworks	10	py
Total	25	py

Proofs and Libraries

Proofs and Libraries

Main Proof Components



Theory Inclusion Graph

Theory Inclusion Graph

Main Libraries

- Libraries
  - Word library
  - Enums
  - Haskell library support
  - Monads
  - Monad VCG + case splitter + strengthening
  - Refinement on Monads
  - Submonads and rewriting under refinement
  - Refinement between Monads and C + tools
  - SIMPL + imperative VCG
  - Crunch
  - LemmaBucket

Main Libraries

- Libraries
  - Word library
  - Enums
  - Haskell library support
  - Monads
  - Monad VCG + case splitter + strengthening
  - Refinement on Monads
  - Submonads and rewriting under refinement
  - Refinement between Monads and C + tools
  - SIMPL + imperative VCG
  - Crunch
  - LemmaBucket

Main Libraries

- Libraries
  - Word library
  - Enums
  - Haskell library support
  - Monads
    - Monad VCG + case splitter + strengthening
    - Refinement on Monads
    - Submonads and rewriting under refinement
  - Refinement between Monads and C + tools
  - SIMPL + imperative VCG
  - Crunch
  - LemmaBucket

Tools

- Productivity tools
  - theorem search (find_thms)
  - theorem moving (Levity)
  - finding definitions (locate)
  - theorem web search (www_find)
  - numeral syntax (hex, oct, binary)

- Isabelle
  - Record package
  - Proof cache
  - attributes like “rotated”
  - automated simp rules for case construct

Tools

- Productivity tools
  - theorem search (find_thms)
  - theorem moving (Levity)
  - finding definitions (locate)
  - theorem web search (www_find)
  - numeral syntax (hex, oct, binary)
- Isabelle
  - Record package
  - Proof cache
  - attributes like “rotated”
  - automated simp rules for case construct

What we thought would work

- Automation
  - First order provers (vampire)
  - SAT for word problems
  - Termination proofs

- What we thought might be a problem
  - Performance
  - Memory
  - Lemmas/goals/proofs too big to parse

Problems

Problems

- What turned out to be a problem

Problems

- What turned out to be a problem
  - Isabelle updates

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently
  - Image rebuilds, theory merges and context

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently
  - Image rebuilds, theory merges and context
  - Avoiding duplication and wheel reinvention

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently
  - Image rebuilds, theory merges and context
  - Avoiding duplication and wheel reinvention
  - Annoying word proofs

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently
  - Image rebuilds, theory merges and context
  - Avoiding duplication and wheel reinvention
  - Annoying word proofs
  - Developing/maintaining libraries + frameworks

Problems

- What turned out to be a problem
  - Isabelle updates
  - Performance, but not as bad
    - theory merges
    - defining large locales + entering large contexts
    - large records
    - waiting for large goals to be printed
  - Memory, but only recently
  - Image rebuilds, theory merges and context
  - Avoiding duplication and wheel reinvention
  - Annoying word proofs
  - Developing/maintaining libraries + frameworks
    - Hard to get over good enough

Wish List

- Better name space management
- Better performance (time + memory)
- Better parallelisation
  - run nitpick, quickcheck, sledgehammer etc in background
  - run proofs faster interactively
- Better dependency handling
  - faster turnaround for deep definition changes
  - faster image rebuilding

Archive of Formal Proofs

Archive of Formal Proofs

AFP

- Repository of Isabelle proofs
  - Open Source (BSD + LGPL)
  - Maintained
  - Archived, version controlled
  - Quality controlled

- Hosted on SourceForge
  - [http://afp.sf.net]

Screen Shot

NICTA

The Archive of Formal Proofs

http://afp.sourceforge.net/

Google

Google SMH Spiegel Slashdot L4v Wiki Bugzilla ERTOS 2 Wiki

The Archive of Formal Proofs

The Archive of Formal Proofs is a collection of proof libraries, examples, and larger scientific developments, mechanically checked in the theorem prover Isabelle. It is organized in the way of a scientific journal and has an ISSN: 2150-914x. Submissions are refereed. The preferred citation style is available [here]. A development version of the archive is available as well.

- Home
- About
- Submission Guidelines
- Updating entries
- Search
- Index
- Download

2010

2010-06-24: Free Groups
Author: Joachim Breitner

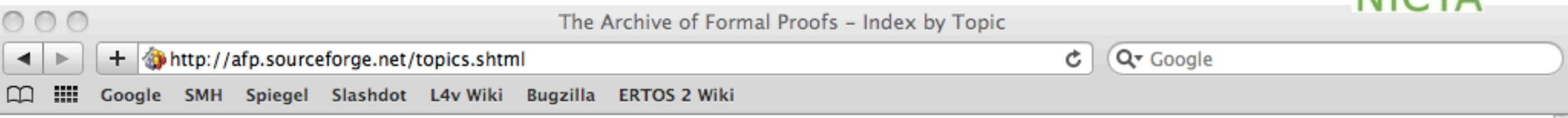
2010-06-20: Category Theory
Author: Alexander Katovsky

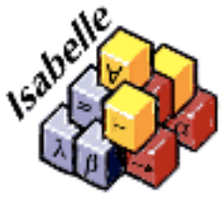
2010-06-17: Executable Matrix Operations on Matrices of Arbitrary Dimensions
Author: Christian Sternagel and René Thiemann

2010-06-14: Abstract Rewriting
Author: Christian Sternagel and René Thiemann

2010-05-28: Verification of the Deutsch-Schorr-Waite Graph Marking Algorithm using Data Refinement
Author: Viorel Preoteasa and Ralph-Johan Back

Index by Topic





INDEX BY TOPIC

- Home
- About
- Submission Guidelines
- Updating entries
- Search
- Index
- Download

Computer Science

Automata and Formal Languages
Regular-Sets Presburger-Automata Functional-Automata Tree-Automata

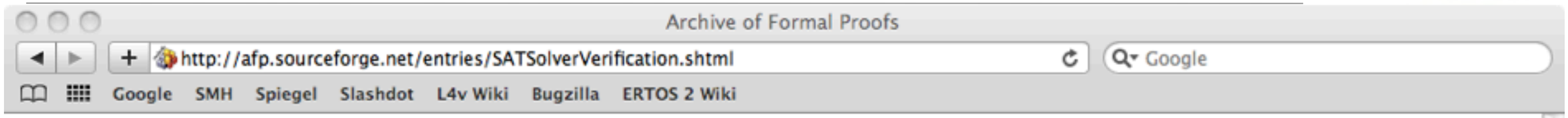
Algorithms
DPT-SAT-Solver Depth-First-Search FFT GraphMarkingIBP
SATSolverVerification MuchAdoAboutTwo **Distributed:** DiskPaxos
GenClock ClockSynchInst

Data Structures
AVL-Trees BDD BinarySearchTree FinFun Collections FileRefinement
List-Index Matrix Huffman Lazy-Lists-II

Functional Programming
Coinductive Stream-Fusion

Programming Languages
Language Definitions: CoreC++ FeatherweightJava Jinja JinjaThreads
Locally-Nameless-Sigma POPLmark-deBruijn Simpl **Type**
Systems: MiniML VolpanoSmith **Logics:** Simpl Abstract-Hoare-Logics
BytecodeLogicJmlTypes DataRefinementIBP SIFPL **Compiling:** Compiling-

An Entry



Formal Verification of Modern SAT Solvers

Title:	Formal Verification of Modern SAT Solvers
Author:	Filip Maric
Submission date:	2008-07-23
Abstract:	This document contains formal correctness proofs of modern SAT solvers. Following (Krstic et al, 2007) and (Nieuwenhuis et al., 2006), solvers are described using state-transition systems. Several different SAT solver descriptions are given and their partial correctness and termination is proved. These include: • a solver based on classical DPLL procedure (using only a backtrack-search with unit propagation), • a very general solver with backjumping and learning (similar to the description given in (Nieuwenhuis et al., 2006)), and • a solver with a specific conflict analysis algorithm (similar to the description given in (Krstic et al., 2007)). Within the SAT solver correctness proofs, a large number of lemmas about propositional logic and CNF formulae are proved. This theory is self-contained and could be used for further exploring of properties of CNF based SAT algorithms.

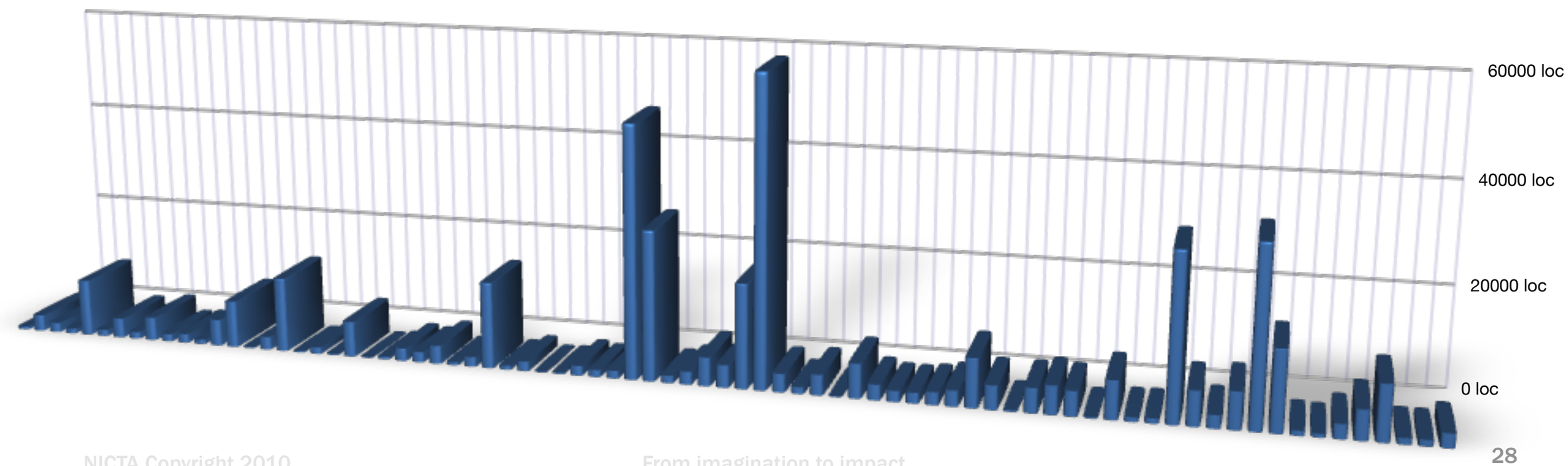
- Home
- About
- Submission Guidelines
- Updating entries
- Search
- Index
- Download

SOURCEFORGE.NET
project summary

Proof outline
Proof document
Browse theories
Download this entry

Statistics

- Numbers
  - started 2003
  - now 77 entries
  - 473 kloc of proof scripts
  - 22,000 lemmas
  - runs 4-6h daily
- Typical entry
  - belongs to a paper
  - nice proof document
  - small, 1-4 kloc
  - some big developments



What works well

- Archival and Maintenance
  - everything kept up to date by Isabelle team
  - new release with every Isabelle release
  - good number of submissions
  - mostly good quality submissions
  - some authors keep improving their entries
    - (but need better high-level change tracking, versioning)

  - good test bed for Isabelle team
  - good place to publish proofs for paper authors
  - authors can get repository access

Reuse?

- Not much reuse:
  - Few entries designed as libraries
  - Few entries used in others (3)

- Reasons
  - Cultural (mostly papers)
  - Good libraries go directly into Isabelle distribution
  - Technical hurdles
    - not enough instruction on how to include
    - could give more built-in support, name spaces again
    - can’t import from more than one image
    - dependencies? hackage site?

Summary

Summary

Summary

Proof Libraries are like Program Libraries

- **L4.verified:**
  - 200kloc proof
  - lots of libraries
  - some are reusable, costs more work, usually worth it

- **Archive of Formal Proofs**
  - 77 entries, 470kloc proof in total
  - works well for papers
  - not so well for libraries yet, hope to get more in the future

NICTA

Thank You