Implementing a portfolio-solving tactic for Why3

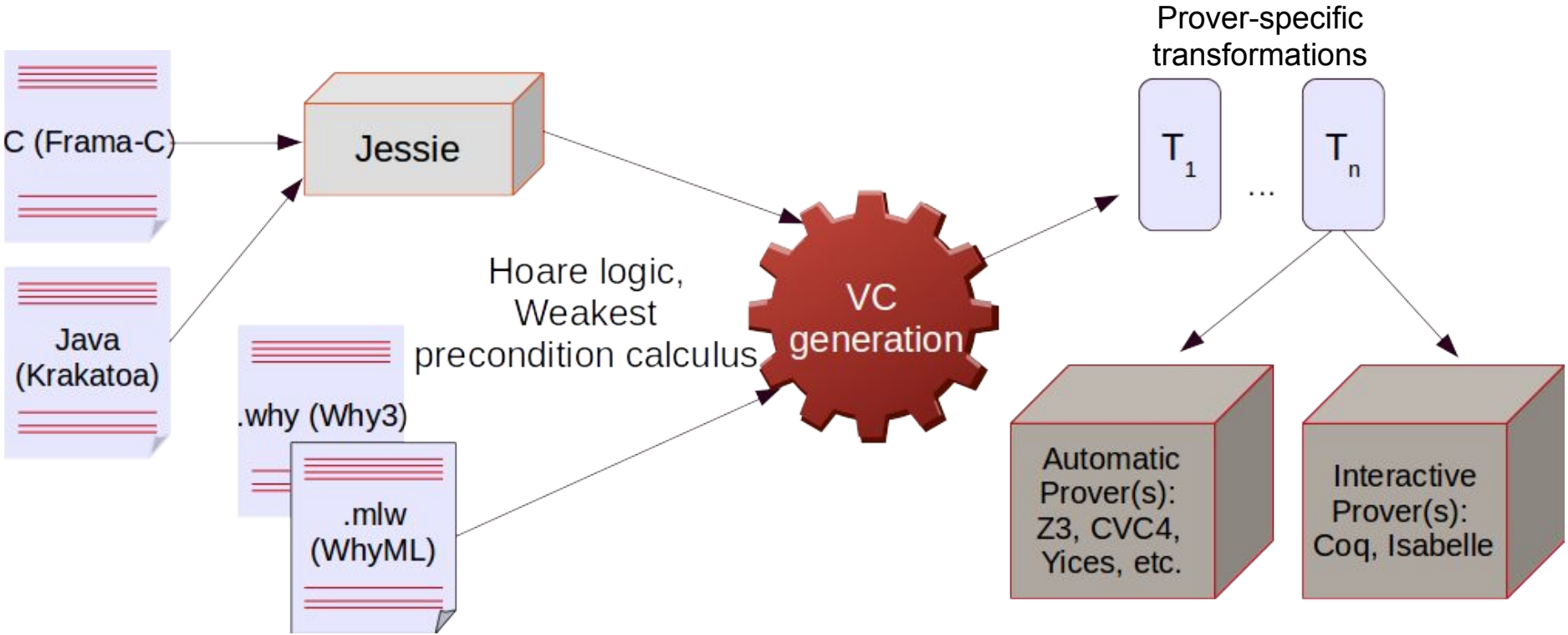
Andrew Healy, Dr. Rosemary Monahan, Dr. James F. Power
Dept. of Computer Science, Maynooth University
Postgraduate Workshop, 15th March 2016

Can Why3 be trained to delegate proof tasks efficiently?

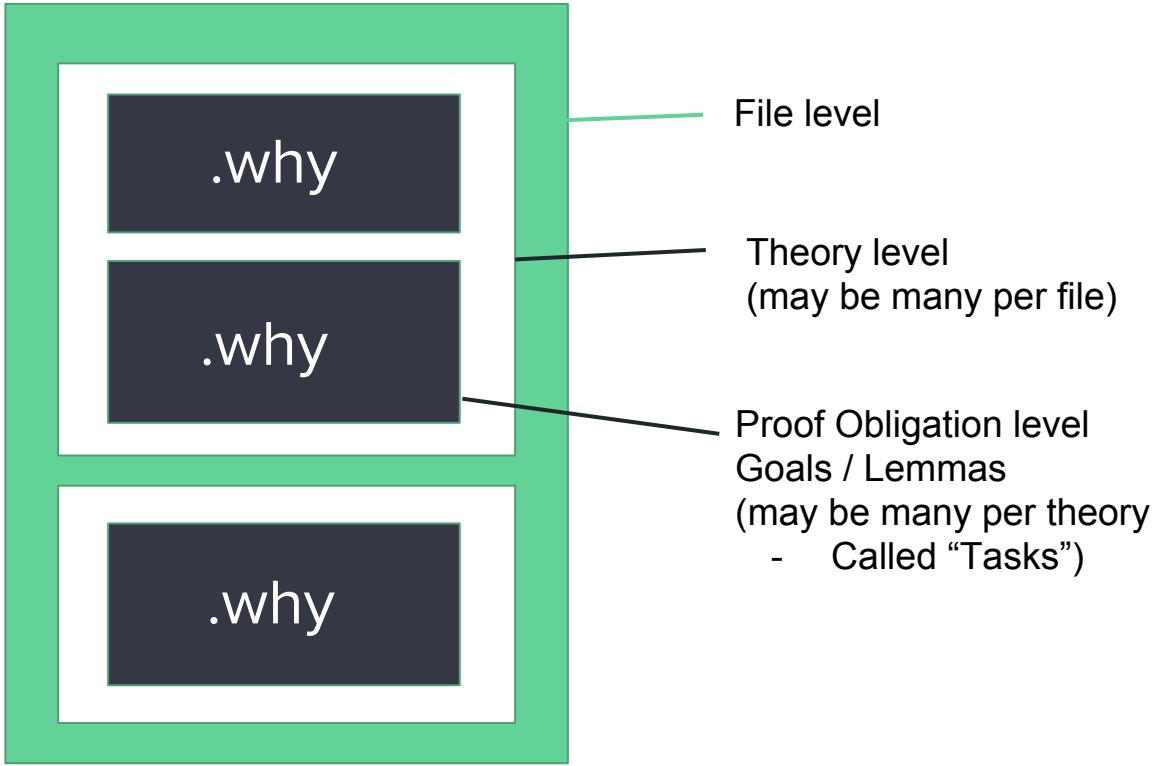
Outline

1. (Re)-introducing Why3
2. Describing the problem
3. Profiling Workload
4. Related Approaches + the Future

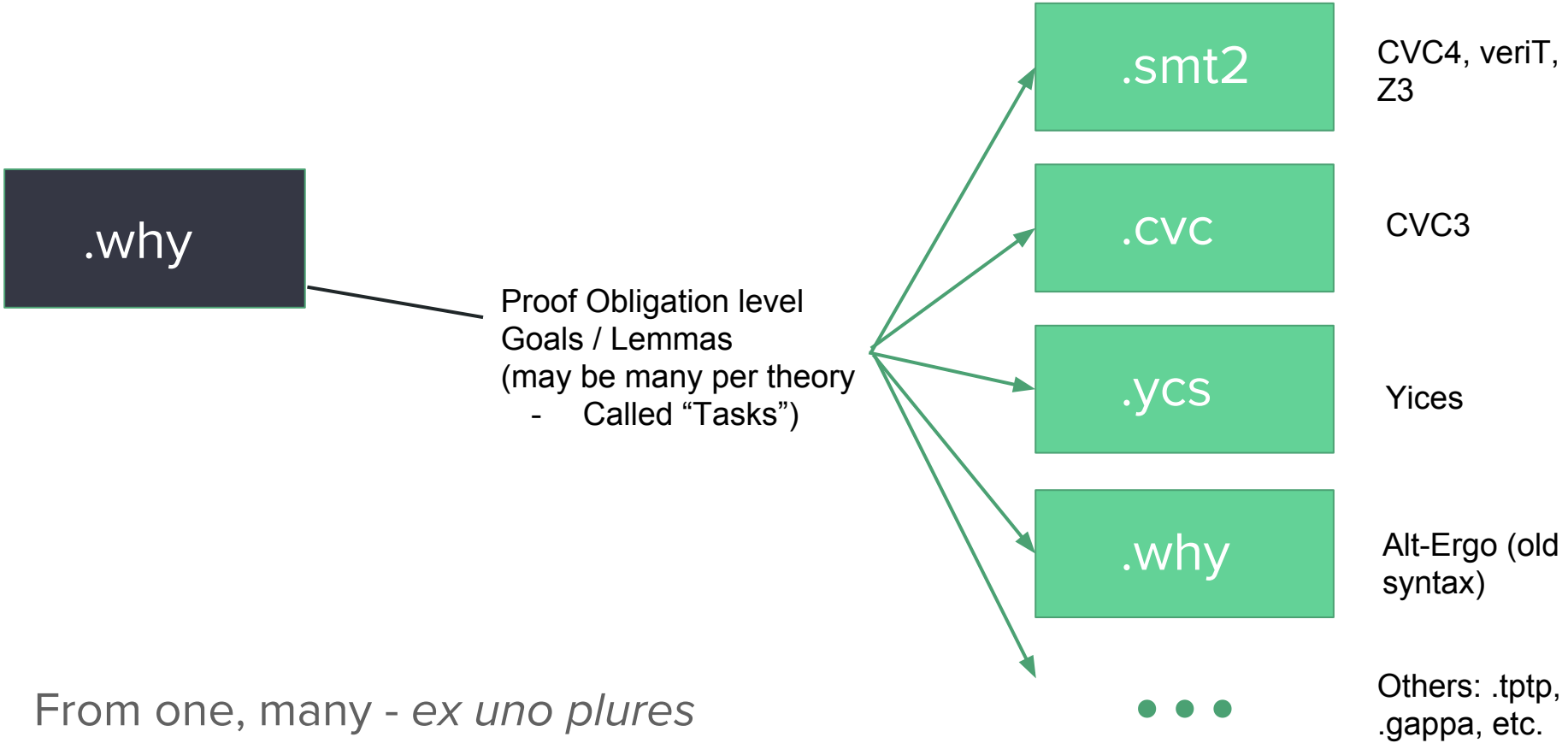
1. (Re)-Introducing Why3



Last Year: Why3 Architecture and Workflow

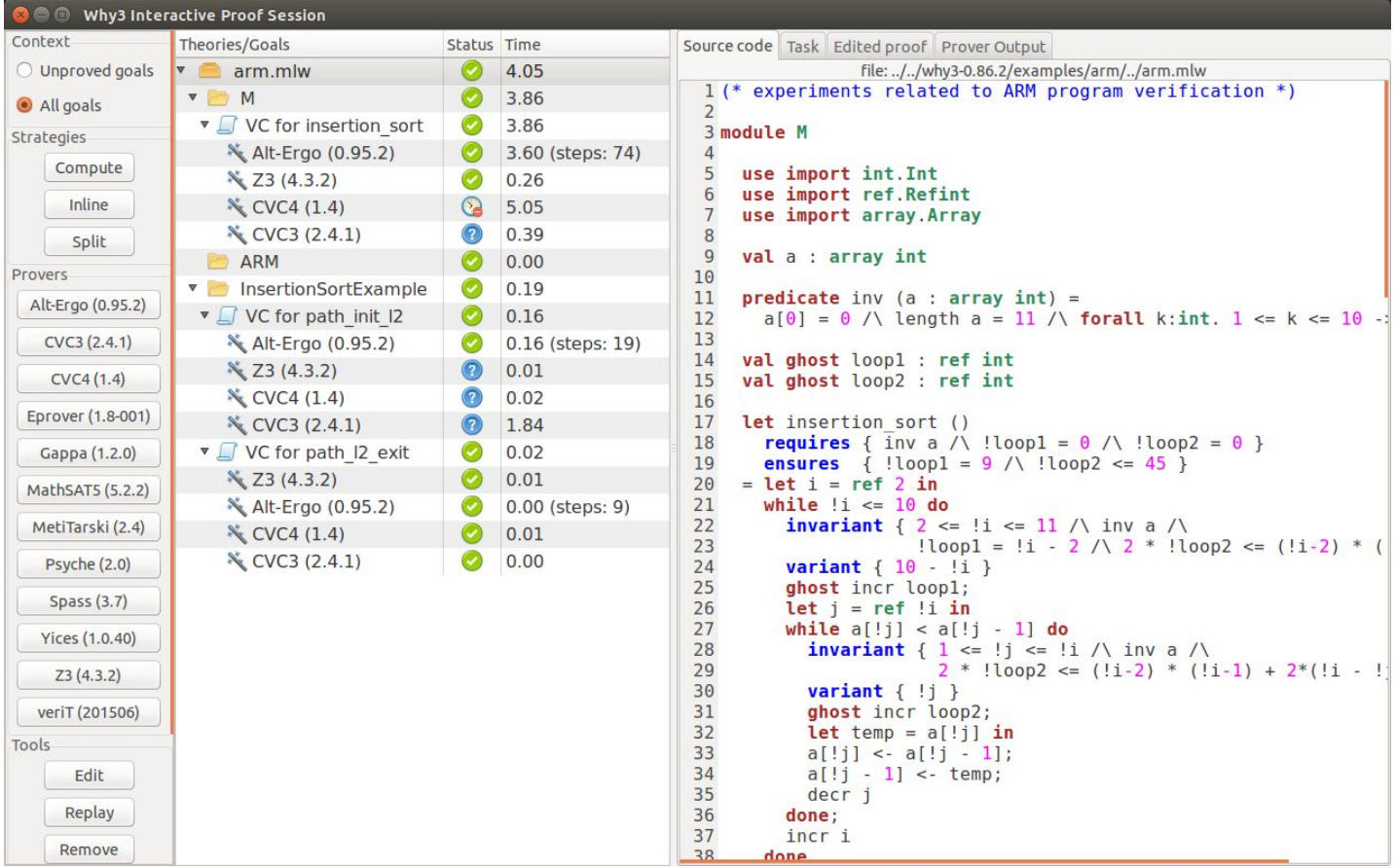


Anatomy of a Why3ML file



From one, many - *ex uno plures*

2. Describing the Problem



What makes one prover better than another?

The problem:

- Difficulty in comparing SMT provers due to the diverse range of input languages and logical specialisations [1]
- We need some way of categorising programs in order to reason about provers
- Knowledge of each prover’s strengths and limitations is needed in order to verify programs

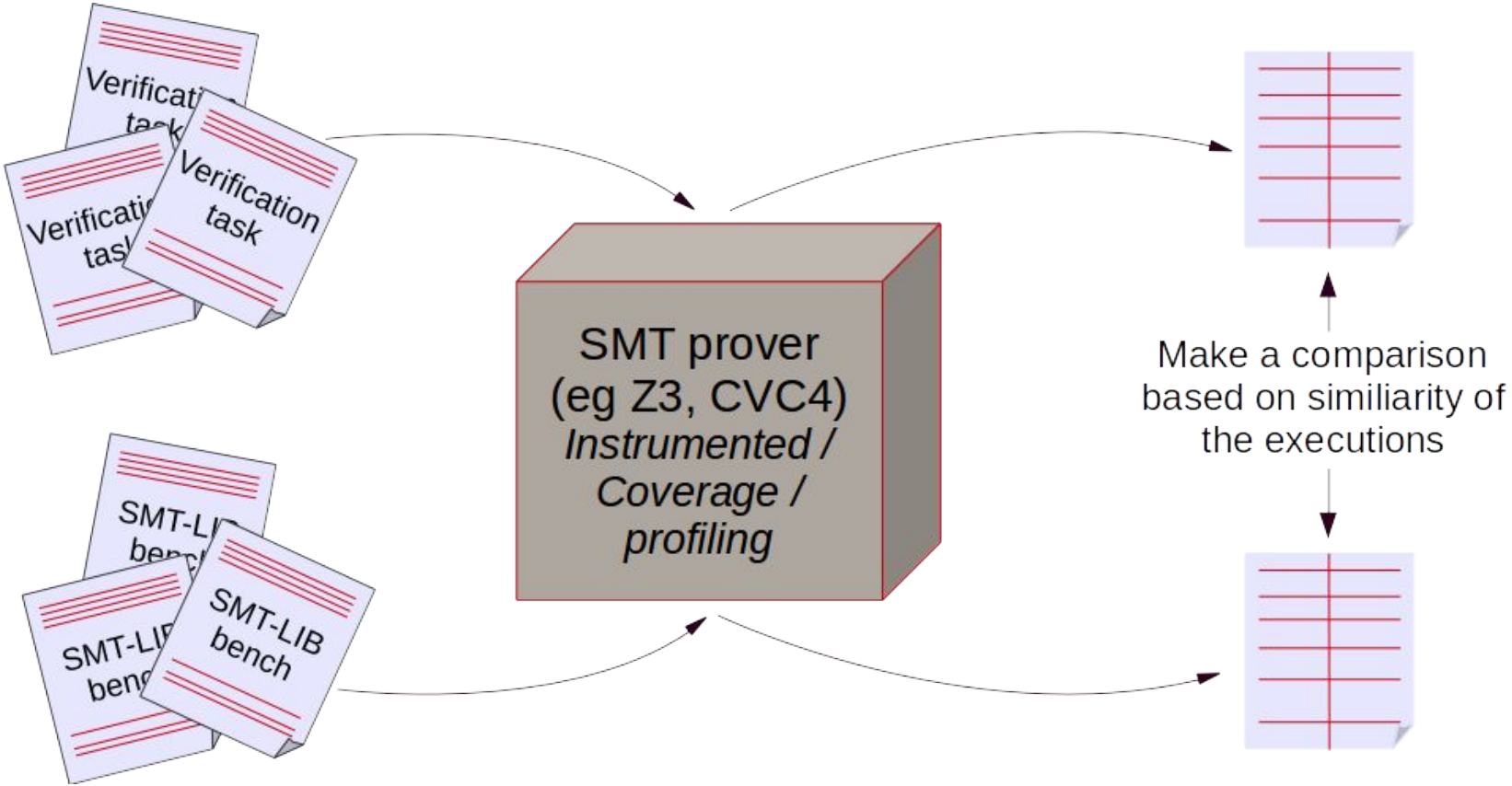
The problem:

- Difficulty in comparing SMT provers due to the diverse range of input languages and logical specialisations [1]
- We need some way of categorising programs in order to reason about provers
- Knowledge of each prover’s strengths and limitations is needed in order to verify programs

A solution?

Use techniques from data analysis and machine learning to predict the most successful prover for a given problem

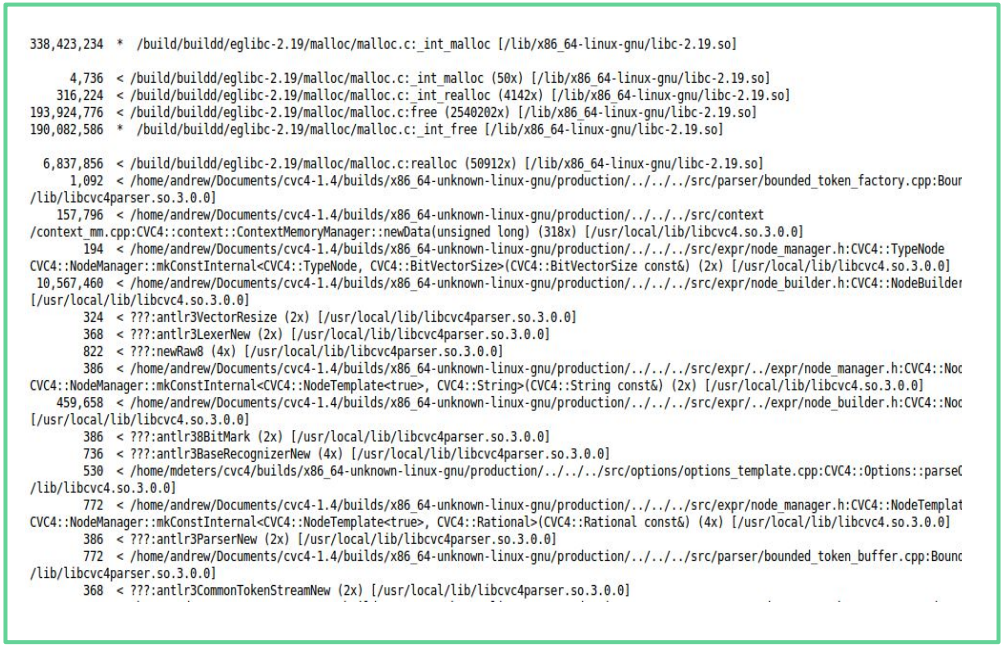
3. Profiling Workload



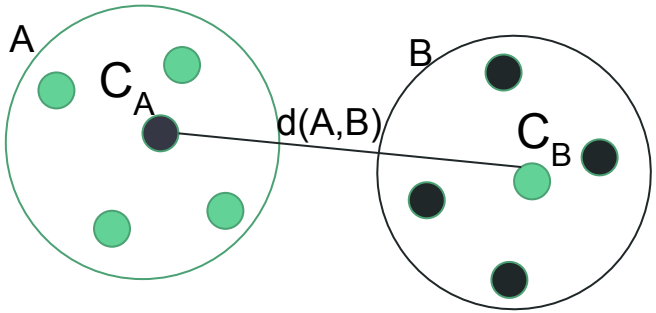
Last Year: A Proposed Workflow

Actual Workflow:

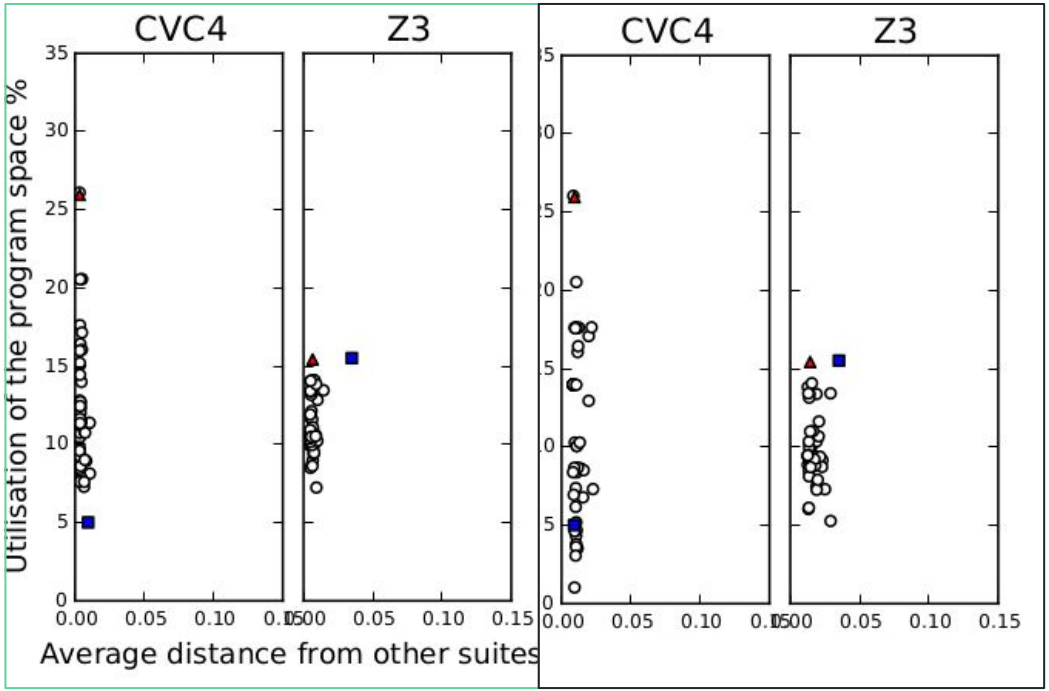
1. Profile programs from the Why3 examples and SMTLIB repo with “callgrind”
2. Filter system calls to maintain platform independence
3. Perform dimensionality reduction via PCA
4. Assuming the Why3 programs are a representative “verification cluster”, measure its distance:
    a. To randomly-sampled SMTLIB programs
    b. To sets of SMTLIB programs composed according to the inherent hierarchy
    c. To a workload-based clustering of SMTLIB programs



Distance Measurement & Results



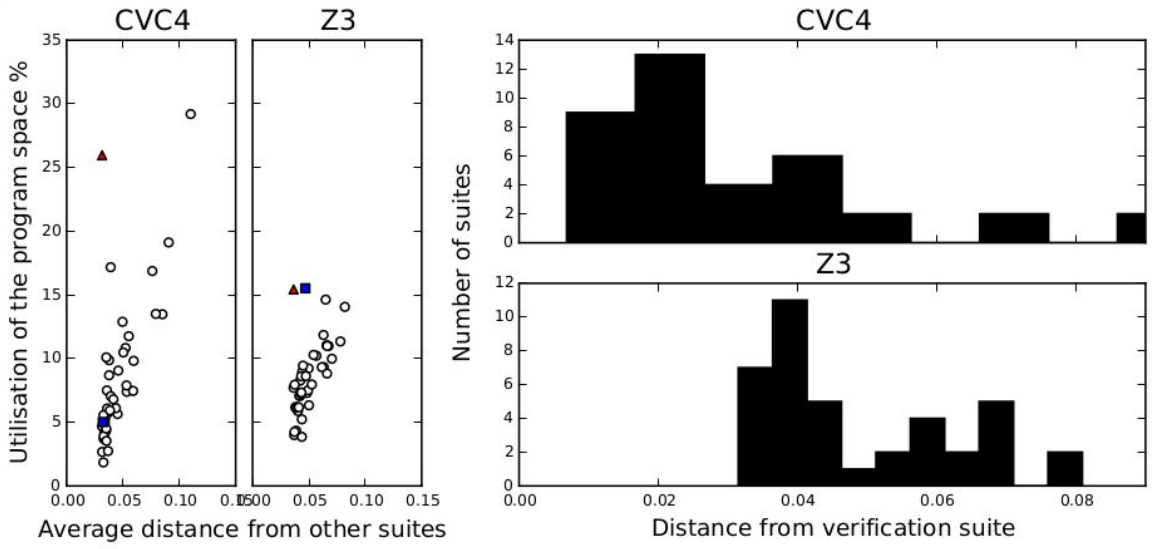
We used techniques and metrics described by Jozo Dujmovic in [2] to accurately characterise and describe machine-independent workloads.



$$diff(A,B) = \frac{1}{2}\sum_{i=0}^{N}\left|p_i^{(A)} - p_i^{(B)}\right|, sim(A,B) = 1 - diff(A,B)$$

What we learned

- Z3 uses more of its features during software verification tasks than CVC4 (experiment 1)
- The SMTLIB repository's structure is more closely aligned to how CVC4 functions (experiment 2)
- A workload-based clustering is more successful at identifying programs that are different from verification programs in the SMTLIB repository than those that are very similar



More details in [3]. Online data: **http://www.cs.nuim.ie/~ahealy/SAC2016**

Taking into account the result of the prover ...

... and the time the prover took to return a result...

Why3 presents an opportunity to combine all this relevant information to produce a useful tool.

4. Related Approaches

Some Portfolio Solvers and ML tools already exist

- **Sledgehammer** for the Isabelle/HOL interactive theorem prover - a Naive Bayes algorithm filters (previously-proved) facts for relevance, guiding interactive proving [3]
- **ML4PG** (Machine Learning for Proof General) Interfaces Coq proofs developed in Proof General to MATLAB/Weka clustering algorithms, again guiding interactive proofs [4]
- **Verifolio** extracts static metrics from C programs to choose the most likely ATP to use (via Support Vector Machines) [5]

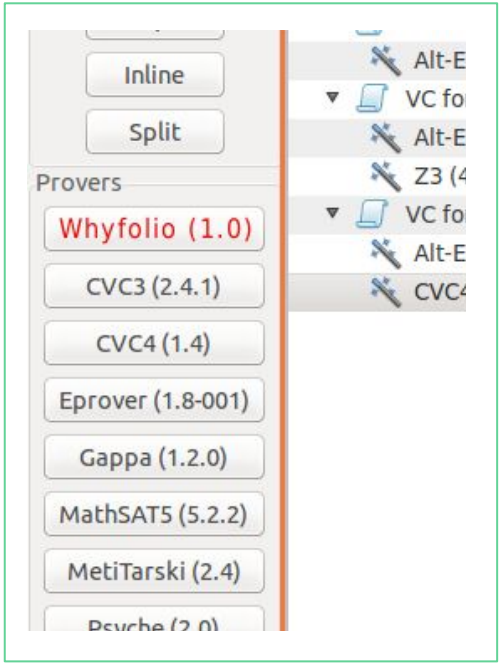
A Portfolio Solver for Why3

Two common input languages, three different ‘levels’ of viewing a program and an extensible driver-based architecture present an opportunity to develop a portfolio solver for SMT-based deductive program verification.

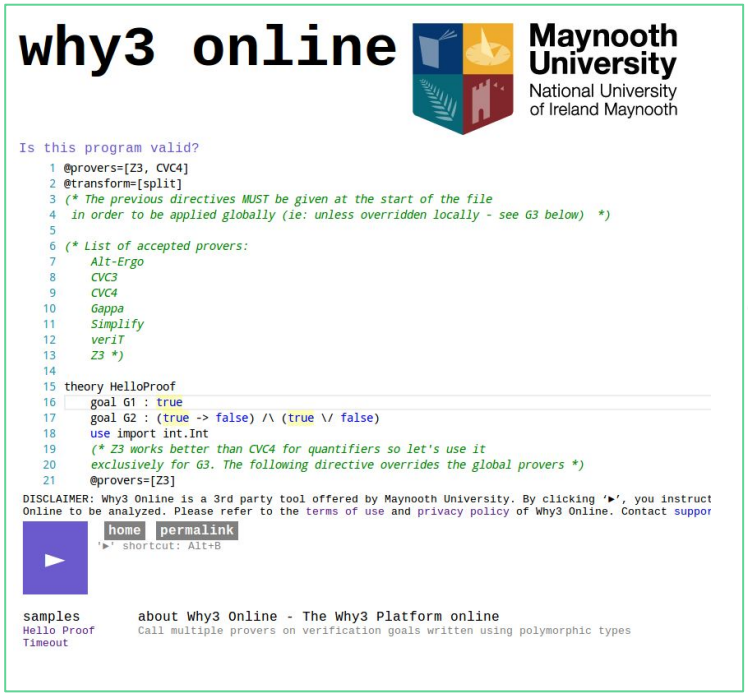
Current work: Implementing a version of process described in [5] through three experiments:

1. Using metrics derived from WhyML programs (contracts, invariants, quantifiers, shape analysis)
2. Using metrics derived from individual proof obligations (why3 logic syntax: predicate logic operators)
3. Comparing predictions from 1+2 to those obtained from instrumentation [3]

As a Why3 plugin



On the Cloud



In a container

Contexts for our tool...

Thanks!

Any questions...

ahealy@cs.nuim.ie

This material is based on works carried out with funding provided by the Science Foundation Ireland under grant number 11/RFP.1/CMS/3068

References

1. D. Beyer, M. Huisman, V. Klebanov, & R. Monahan. *Evaluating Software Verification Systems: Benchmarks and Competitions* (Dagstuhl Reports 14171). **Dagstuhl Reports**, 4(4):1–19, 2014

2. J. Dujmovic. *Universal benchmark suites - a quantitative approach to benchmark design*. In R. Eigenmann, ed., **Performance Evaluation and Benchmarking with Realistic Applications**. MIT Press, 2001.

3. A. Healy, R. Monahan, J. Power. *Evaluating the Use of a General-Purpose Benchmark Suite for Domain-Specific* SMT-solving, Proceeding of **Symposium on Applied Computing**, 2016.

4. D. Kuhlwein, J. C. Blanchette, C. Kaliszyk & J. Urban. *MaSh: Machine Learning for Sledgehammer.* In: 4th Conference on Interactive Theorem Proving (ITP'13), **Lecture Notes in Computer Science**.

5. E. Komendantskaya, J. Heras & G. Grov. *Machine Learning for Proof General: interfacing interfaces.* EPTCS Post-proceedings of **User Interfaces for Theorem Provers** (UITP 2012) see also ai4fm.org

6. Y. Demyanova, T. Pani, H. Veith, and F. Zuleger. *Empirical software metrics for benchmarking of verification tools*. In Computer Aided Verification, vol. 9206 of **Lecture Notes in Computer Science**.