Erasusmus Mundus Summer School
30th June 2015

Evaluating the SMT-LIB repository as a benchmark source for software verification

Andrew Healy, MSc Computer Science (by Research)
Maynooth University, Ireland

Supervisors: Dr Rosemary Monahan & Dr James F. Power

Introduction

- **This project looks at the SMT solving component found in many verification systems**

Introduction

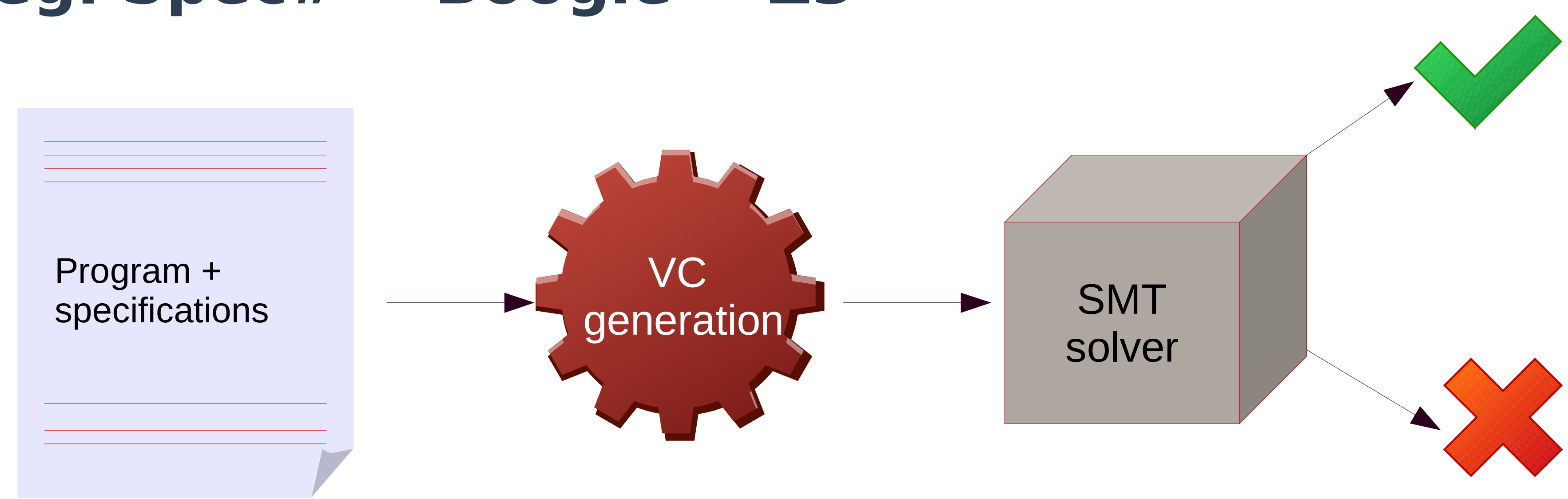
- **This project looks at the SMT solving component found in many verification systems**
- **Often the components in these systems are tightly integrated and optimised for efficiency**

Introduction

- **This project looks at the SMT solving component found in many verification systems**
- **Often the components in these systems are tightly integrated and optimised for efficiency**
- **eg. Spec# → Boogie → Z3**

Introduction

- **This project looks at the SMT solving component found in many verification systems**
- **Often the components in these systems are tightly integrated and optimised for efficiency**
- **eg. Spec# → Boogie → Z3**



The need for benchmarks

- **The ecosystem of verification systems that follow this pattern is varied in terms of:**
  - input language

The need for benchmarks

- **The ecosystem of verification systems that follow this pattern is varied in terms of:**
  - input language
  - backend SMT implementation

The need for benchmarks

- **The ecosystem of verification systems that follow this pattern is varied in terms of:**
  - input language
  - backend SMT implementation
  - logical / programming specialisation (if any)

The need for benchmarks

- **The ecosystem of verification systems that follow this pattern is varied in terms of:**
  - input language
  - backend SMT implementation
  - logical / programming specialisation (if any)

- **A common suite of programs would aid qualitative comparison of verification systems** *(Beyer et al, 2014)*

The need for benchmarks

Benchmarks are a set of agreed and shared applications, with which we evaluate and compare computer systems as well as their hardware and software components. Hence, benchmarks make it possible to compare results on a common basis, which is a most important quality of all academic research.

– *Rudolf Eigenmann*

A brief overview of SMT

- **Satisfiability Modulo Theories**

A brief overview of SMT

- **Satisfiability Modulo Theories**
- **Extend SAT by combining theories through the Nelson-Oppen procedure (usually)**

A brief overview of SMT

- **Satisfiability Modulo Theories**
- **Extend SAT by combining theories through the Nelson-Oppen procedure (usually)**
- **If a *decision procedure* exists for a theory, it can be implemented by a SMT solver to increase its expressive power**

A brief overview of SMT

- **Satisfiability Modulo Theories**
- **Extend SAT by combining theories through the Nelson-Oppen procedure (usually)**
- **If a *decision procedure* exists for a theory, it can be implemented by a SMT solver to increase its expressive power**
- **Examples of theories: bit-vectors, linear integer arithmetic, uninterpreted functions**

A brief overview of SMT

- **Satisfiability Modulo Theories**
- **Extend SAT by combining theories through the Nelson-Oppen procedure (usually)**
- **If a *decision procedure* exists for a theory, it can be implemented by a SMT solver to increase its expressive power**
- **Examples of theories: bit-vectors, linear integer arithmetic, uninterpreted functions**
- **Have found applications beyond verification: scheduling, modelling and static analysis** *(De Moura et al, 2011)*

The SMT-LIB project

- **Consists of a standard input language, benchmark repository and annual competition (SMT-COMP)**

The SMT-LIB project

- **Consists of a standard input language, benchmark repository and annual competition (SMT-COMP)**
- **Importantly, these three aspects are interdependent and their development is related**

The SMT-LIB project

- **Consists of a standard input language, benchmark repository and annual competition (SMT-COMP)**
- **Importantly, these three aspects are interdependent and their development is related**
- **The SMT-LIB has played an important role in facilitating communication among the SMT community, defining the top-performing tools and accelerating tool development** *(Cok et al, 2014)*

SMT-LIB language example

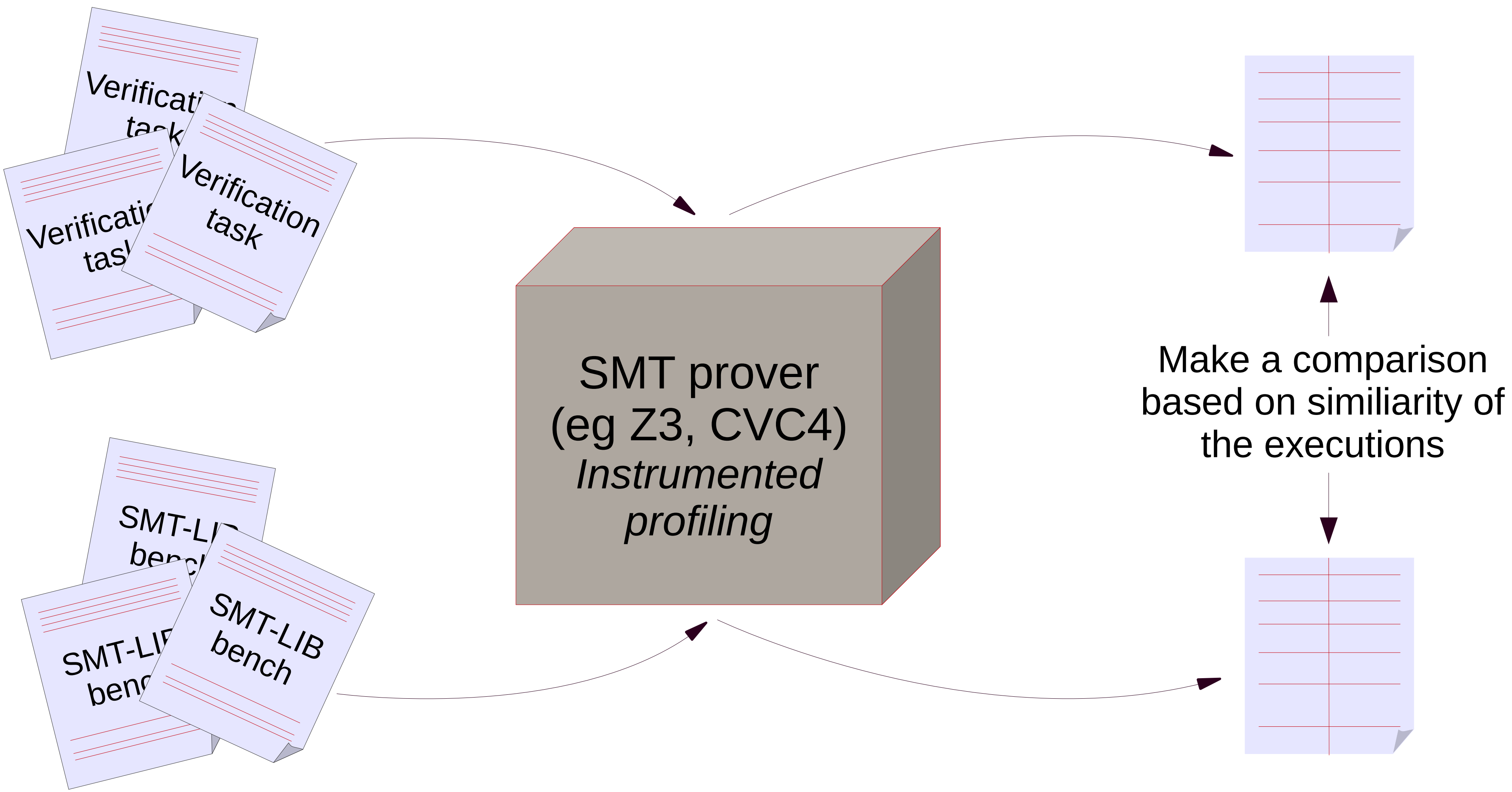
```
> (set-logic QF_LIA)
> (declare-fun x () Int)
> (declare fun y () Int)
> (assert (= (+ x (* 2 y)) 20))
> (assert (= (- x y) 2))
> (check-sat)
sat
> (get-value (x y))
((x 8)(y 6))
```

From David Coq's SMT-LIB v2 Tutorial:
http://www.grammatech.com/resource/smt/SMTLIBTutorial.pdf

Questions

- **Given the large SMTLIB repository of benchmarks (currently over 100,000 programs):**
  - Can we make use of these to test a tool's suitability for use in a software verification context?
  - Is there a workload that is specific to verification tasks?
  - How do verification problems exercise the capabilities of the SMT tool?
  - Does the organisation/structure of the repository reflect features used?

Process overview



Process details 1

Selection of sample verification programs

The programs from the Why3 examples folder that do not require interactive proving (with Coq, for example)

We assume that these programs are representative of the verification problems typically solved by SMT tools

Process details 2

Selection of SMT tools

(constraint: must support SMT-LIB standard version 2 fully; must have Why3 smt2 driver)

- **CVC4** (NYU & Uni. of Iowa)
- **Z3** (Microsoft Research)

Process details 3

Use of Why3 drivers for conversion to SMT-v2

Separates each program (in the .mlw format) into separate goals (in the .smt2 format) to be proved by the solver

Works via a solver-specific series of transformations

Command:

```
why3 prove -D <driver> -o <output-folder> <mlw-file>
```

Process details 4

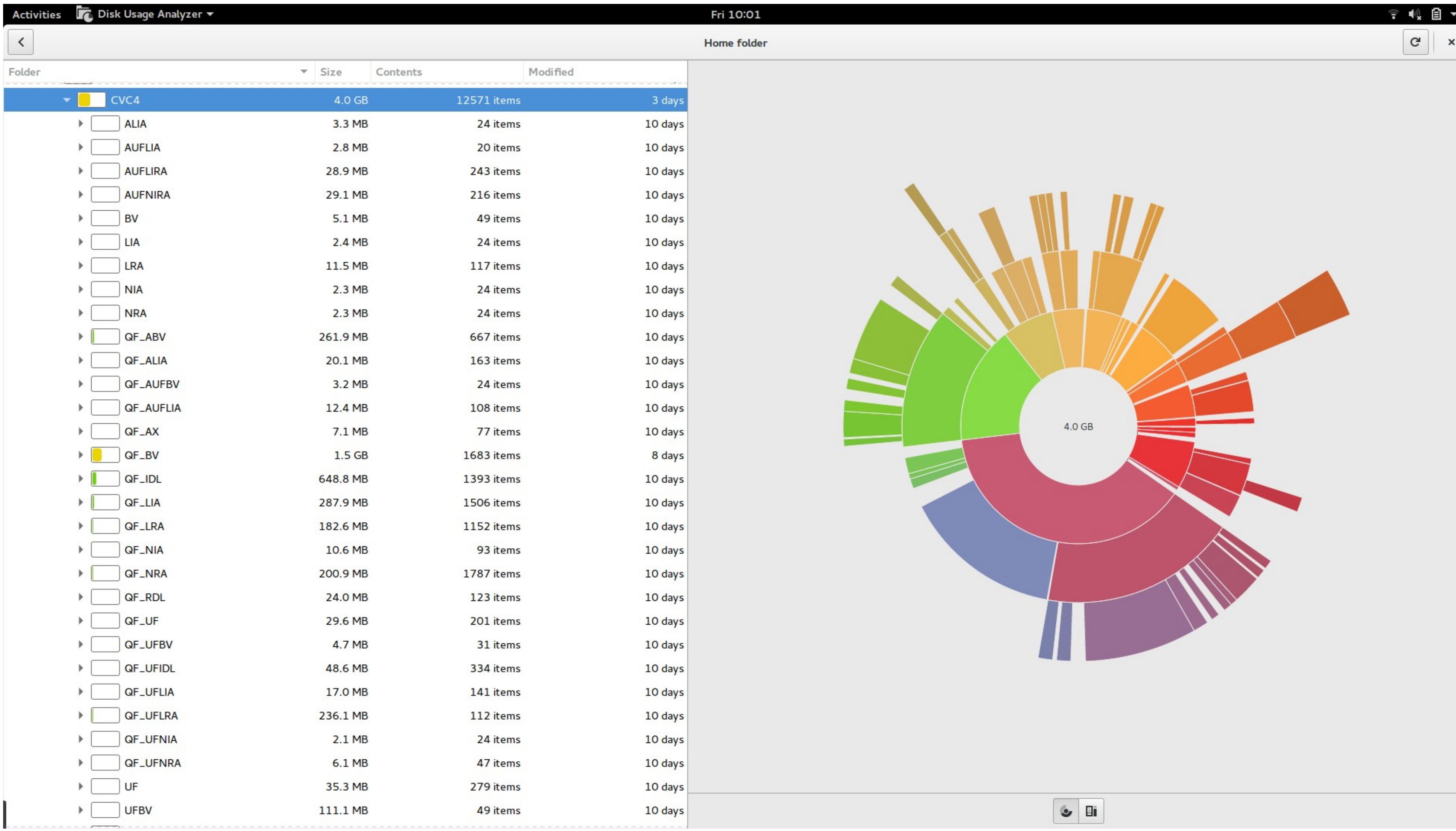
Selection of SMT-LIB benchmarks

Are all 100,000 programs necessary?

A maximum of 5 were chosen from the leaves of the repository

Emphasis on retaining the structure

Process details 4



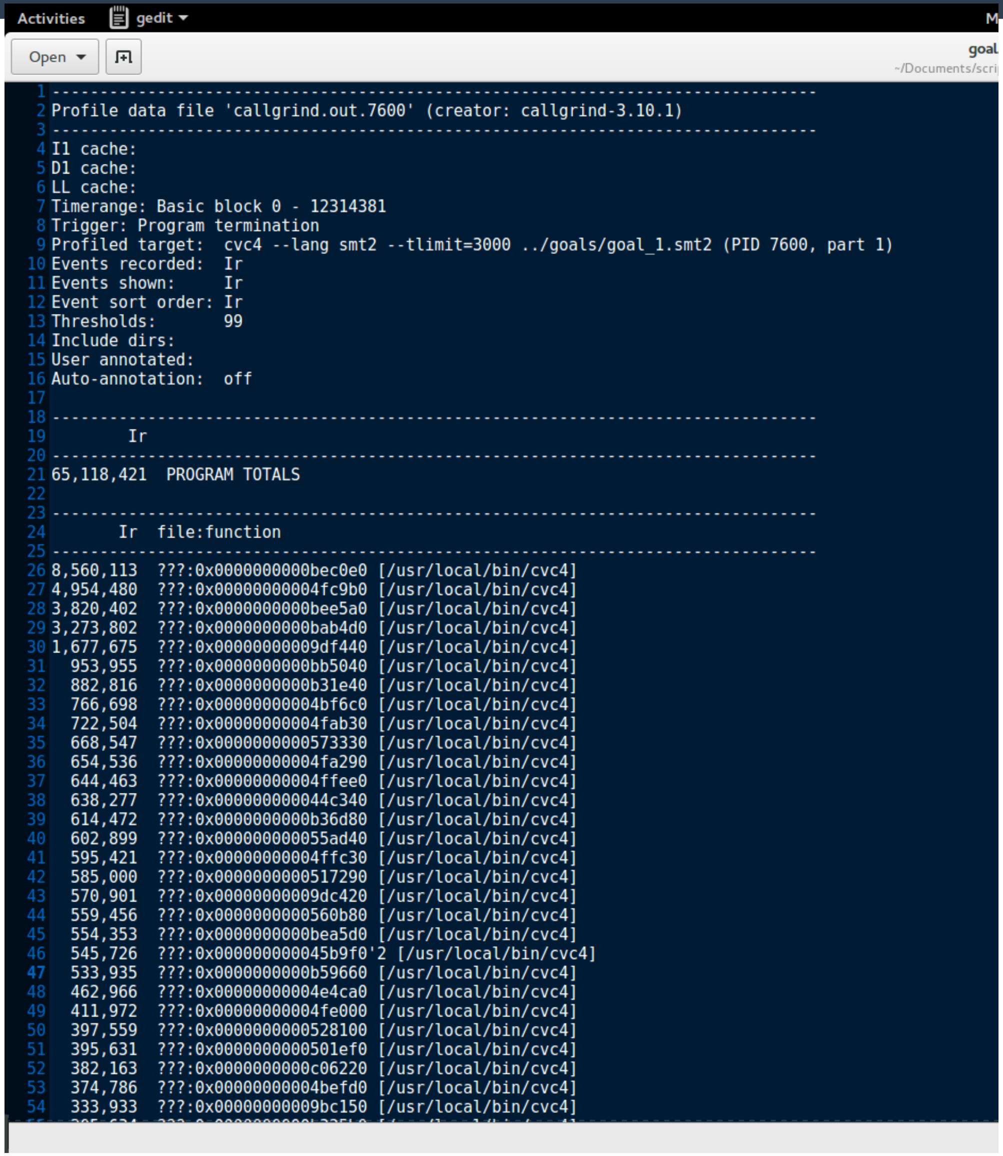
Process details 5

Instrumentation of execution using callgrind

A profiling tool in the valgrind suite that records the number of machine instructions executed by each method

- Slows down execution by 2x – 10x
- A timeout of 60,000ms was enforced for each SMT solver

Process details 5




```
--------------------------------------------------------------------------------
Profile data file 'callgrind.out.7600' (creator: callgrind-3.10.1)
--------------------------------------------------------------------------------
I1 cache:
D1 cache:
LL cache:
Timerange: Basic block 0 - 12314381
Trigger: Program termination
Profiled target:  cvc4 --lang smt2 --tlimit=3000 ../goals/goal_1.smt2 (PID 7600, part 1)
Events recorded:  Ir
Events shown:     Ir
Event sort order: Ir
Thresholds:       99
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
        Ir
--------------------------------------------------------------------------------
65,118,421  PROGRAM TOTALS

--------------------------------------------------------------------------------
       Ir  file:function
--------------------------------------------------------------------------------
8,560,113  ???:0x0000000000bec0e0 [/usr/local/bin/cvc4]
4,954,480  ???:0x00000000004fc9b0 [/usr/local/bin/cvc4]
3,820,402  ???:0x0000000000bee5a0 [/usr/local/bin/cvc4]
3,273,802  ???:0x0000000000bab4d0 [/usr/local/bin/cvc4]
1,677,675  ???:0x00000000009df440 [/usr/local/bin/cvc4]
  953,955  ???:0x0000000000bb5040 [/usr/local/bin/cvc4]
  882,816  ???:0x0000000000b31e40 [/usr/local/bin/cvc4]
  766,698  ???:0x00000000004bf6c0 [/usr/local/bin/cvc4]
  722,504  ???:0x00000000004fab30 [/usr/local/bin/cvc4]
  668,547  ???:0x0000000000573330 [/usr/local/bin/cvc4]
  654,536  ???:0x00000000004fa290 [/usr/local/bin/cvc4]
  644,463  ???:0x00000000004ffee0 [/usr/local/bin/cvc4]
  638,277  ???:0x000000000044c340 [/usr/local/bin/cvc4]
  614,472  ???:0x0000000000b36d80 [/usr/local/bin/cvc4]
  602,899  ???:0x000000000055ad40 [/usr/local/bin/cvc4]
  595,421  ???:0x00000000004ffc30 [/usr/local/bin/cvc4]
  585,000  ???:0x0000000000517290 [/usr/local/bin/cvc4]
  570,901  ???:0x00000000009dc420 [/usr/local/bin/cvc4]
  559,456  ???:0x0000000000560b80 [/usr/local/bin/cvc4]
  554,353  ???:0x0000000000bea5d0 [/usr/local/bin/cvc4]
  545,726  ???:0x000000000045b9f0'2 [/usr/local/bin/cvc4]
  533,935  ???:0x0000000000b59660 [/usr/local/bin/cvc4]
  462,966  ???:0x00000000004e4ca0 [/usr/local/bin/cvc4]
  411,972  ???:0x00000000004fe000 [/usr/local/bin/cvc4]
  397,559  ???:0x0000000000528100 [/usr/local/bin/cvc4]
  395,631  ???:0x0000000000501ef0 [/usr/local/bin/cvc4]
  382,163  ???:0x0000000000c06220 [/usr/local/bin/cvc4]
  374,786  ???:0x00000000004befd0 [/usr/local/bin/cvc4]
  333,933  ???:0x00000000009bc150 [/usr/local/bin/cvc4]
```

Process details 6

Clustering to identify programs with similar workload characteristics

- Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity, but are very dissimilar to objects in other clusters.

Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measures. *(Han 2012)*

Process details 7

The 'curse of dimensionality'

Customer	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
Ada	1	0	0	0	0	0	0	0	0	0
Bob	0	0	0	0	0	0	0	0	0	1
Cathy	1	0	0	0	1	0	0	0	0	1

Euclidean distance:

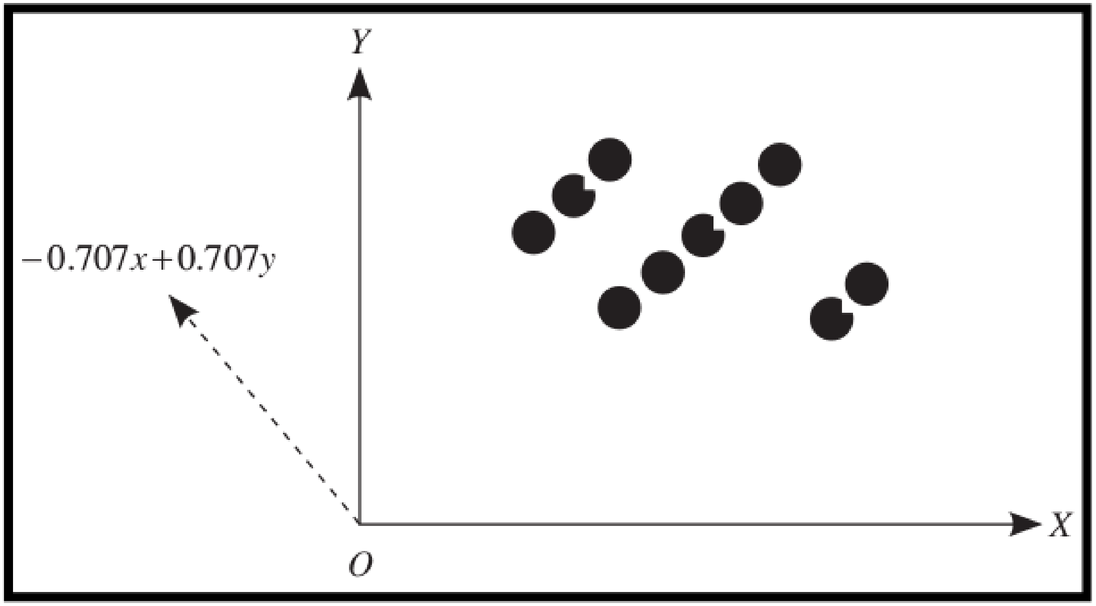
$$dist(\mathrm{Ada}, \mathrm{Bob}) = dist(\mathrm{Bob}, \mathrm{Cathy}) = dist(\mathrm{Ada}, \mathrm{Cathy}) = \sqrt{2}.$$

Makes traditional similarity measures less useful

Process details 7

Dimensionality reduction

- **PCA (Principle Component Analysis)**

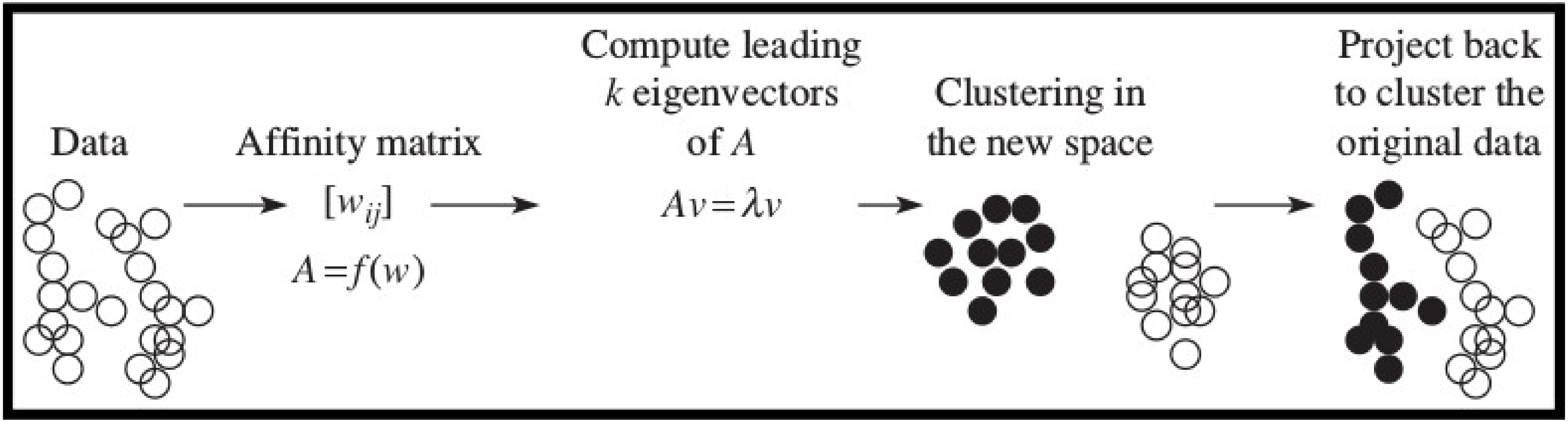


Taken from (Han 2012)

Process details 7

Dimensionality reduction

- **Spectral Clustering (more suitable for sparse, high dimensional matrices)**



Taken from (Han 2012)

Process details 8

Affinity matrices

Jozo Dujmovic presents a domain-specific formula for measuring the similarity of two programs *A* and *B* in terms of time and instruction count for each method *p*.

The platform-independent version assumes time is a constant, giving:

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

(Dujmovic 2001)

Similarity of SMT workloads

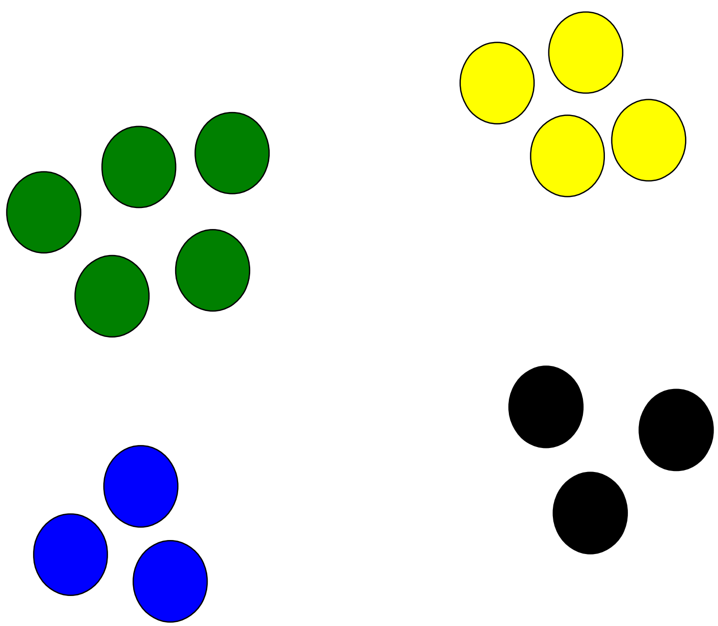
Using the Dujmovic formula for platform independent similarity/measurement:

Verification:

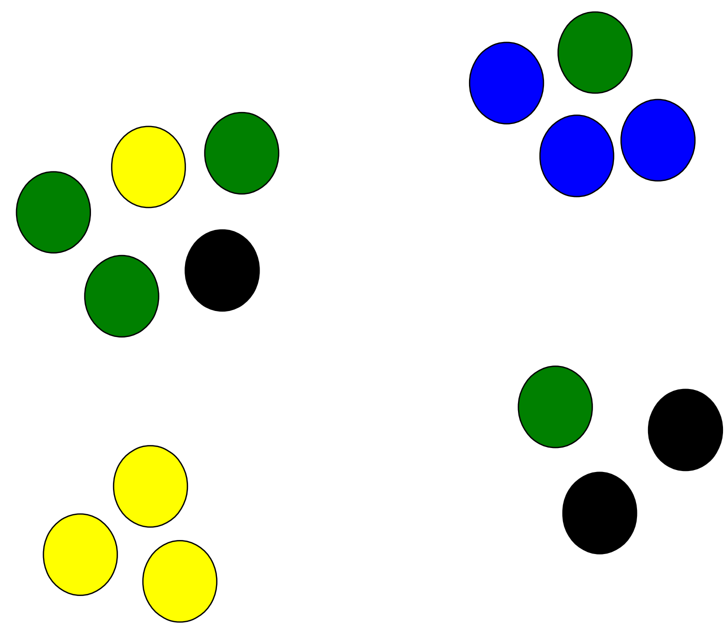
- **Z3:** 0.391710
- **CVC4:** 0.748594

This pattern was repeated in most of across the highest-level SMTLIB subdirectories

Clustering with ground truths



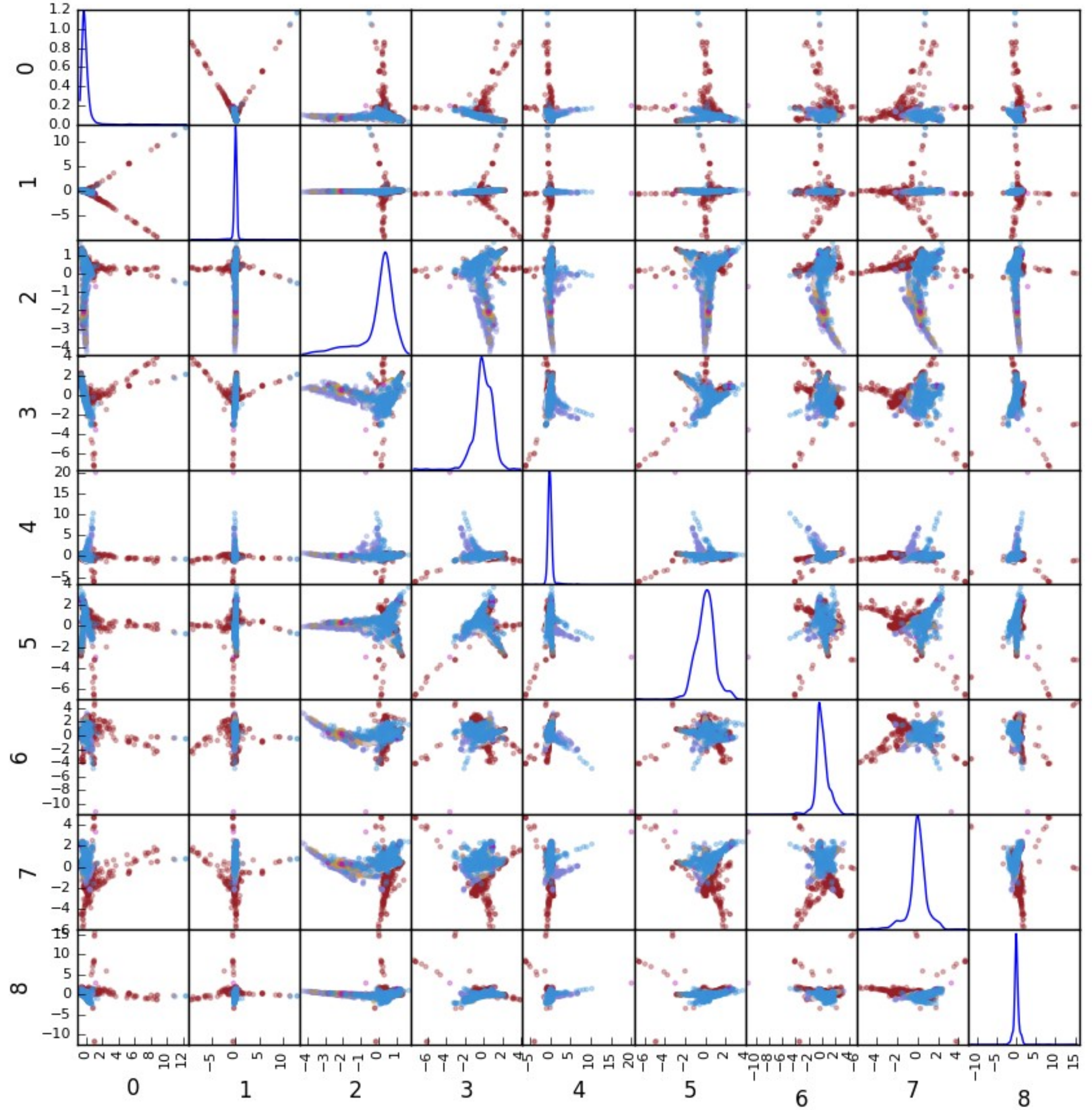
Expected labelling



Actual Clusters

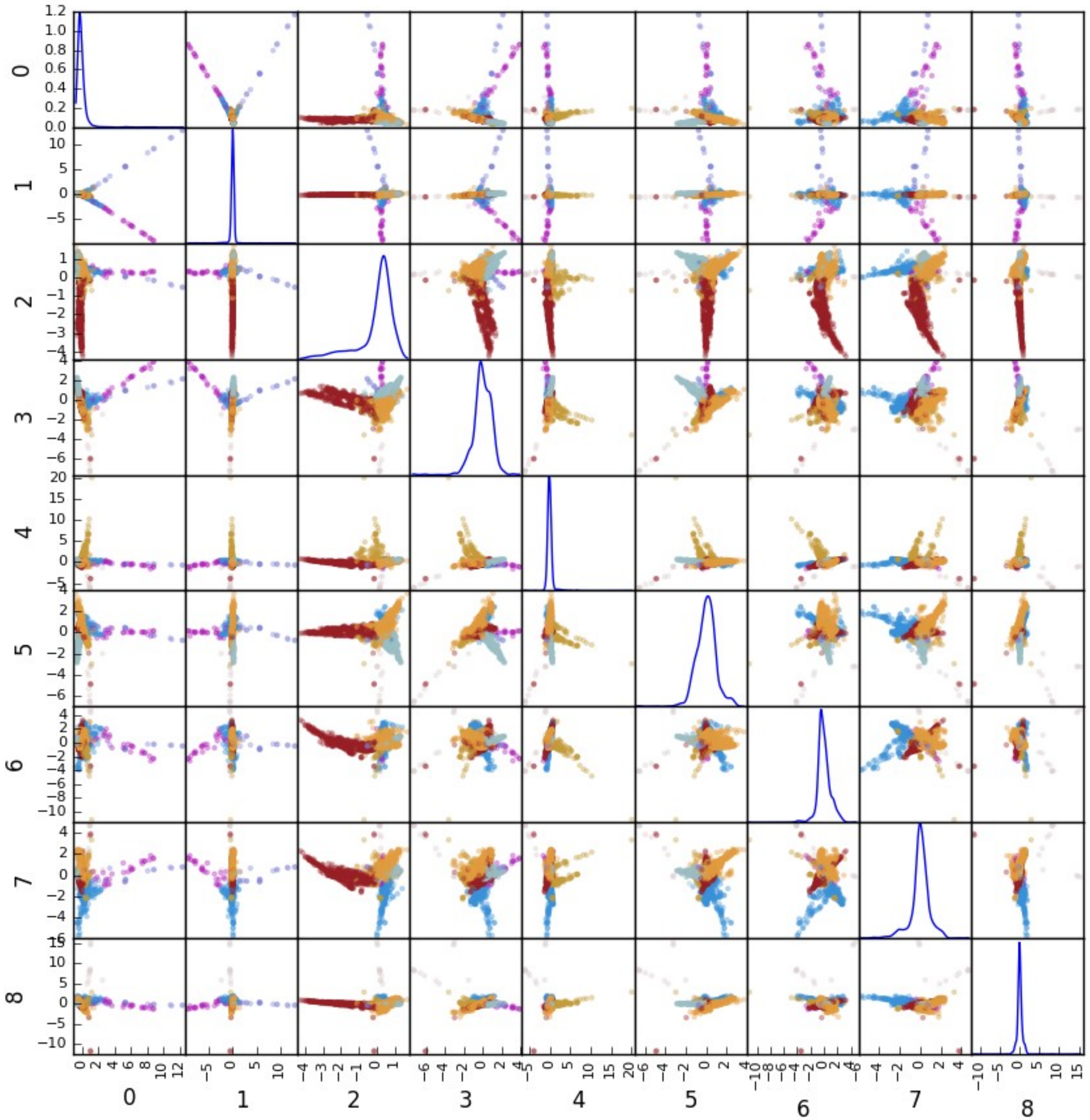
Clustering with ground truths

Actual labelling: CVC4, PCA, agglomerative clustering



Clustering with ground truths

Expected labelling: CVC4, PCA, agglomerative clustering



What we've learned so far

- **The expected labels do not particularly correspond to actual clustering observed** (a best Adjusted Rand Score of ~0.55)
- **Non-negative Factorisation combined with Kmeans can give good results on our high-dimensional, sparse matrices** (a highest silhouette coefficent of ~0.75)
- **Visualising high-dimensional data & clusters with many data points is difficult**

Possible Future Work: integration into Why3 to advise on best transformation to be applied when proving properties with a particular solver

References

- **Dirk Beyer, Marieke Huisman, Vladimir Klebanov, and Rosemary Monahan**. “Evaluating Software Verification Systems: Benchmarks and Competitions” (Dagstuhl Reports 14171). *Dagstuhl Reports*, 4(4):1–19, 2014
- **David R. Cok, Aaron Stump, and Tjark Weber.** “The 2013 SMT evaluation”. Technical Report 2014-017, Department of Information Technology, Uppsala University, July 2014.
- **Leonardo De Moura and Nikolaj Bjørner**. “Satisfiability modulo theories: Introduction and applications”. *Commun. ACM*, 54(9):69–77, September 2011.
- **Jozo Dujmovic.** “Universal Bechmark Suites – A Quantitative Approach to Benchmark Design” in (Eigenmann (ed) 2001)
- **Rudolf Eigenmann**, ed. *Performance Evaluation and Benchmarking with Realistic Applications*. MIT Press, Cambridge MA 2001
- **Jiawei Han**. *Data mining: concepts and techniques* (3rd edition). Morgan Kauffman, 2012

Thank You

andrew.healy.2014@mumail.ie