FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Formal Methods
in Systems Engineering

Empirical Software Metrics for Benchmarking of Verification Tools

Thomas Pani
pani@forsyte.tuwien.ac.at

joint work with
Yulia Demyanova, Helmut Veith, Florian Zuleger

AVM · Attersee · May 5, 2015

Competition on Software Verification (SV-COMP) 2015

John
Software Engineer

Competition on Software Verification (SV-COMP) 2015

Verify my safety-critical code!

tools

competition categories

Competition candidate	AProVE	Beagle	Cascade	CBMC	CPAchecker	CPArec	ESBMC 1.24.1	FORESTER	Forester	FunctTion	HIPTNT+	Lazy-CSeq	Map2Check	MU-CSeq	Perentie	Predator	SeaHorn	SMACK+Corral	Ultimate Automizer	Ultimate Kojak	Unbounded Lazy-CSeq
Representing Jury Member	Thomas Ströder	Dexi Wang	Wei Wang	Michael Tautschnig	Matthias Dangl	Ming-Hsien Tsai	Jeremy Morse	Pablo Sánchez	Ondrej Lengal	Caterina Urban	Ton-Chanh Le	Gennaro Parlato	Herbert Oliveira Rocha	Bernd Fischer	Franck Cassez	Tomas Vojnar	Arie Gurfinkel	Zvonimir Rakamaric	Matthias Heizmann	Alexander Nutz	Salvatore La Torre
Affiliation	Aachen, Germany	Beijing, China	New York, USA	London, UK	Passau, Germany	Taipei, Taiwan	Bristol, UK	Cantabria, Spain	Brno, Czechia	Paris, France	Singapore, Singapore	Southampton, UK	Manaus, Brazil	Stellenbosch, South Africa	Sydney, Australia	Brno, Czechia	Pittsburgh, USA	Salt Lake City, USA	Freiburg, Germany	Freiburg, Germany	Southampton, UK
Arrays 86 tasks, max. score: 145	--	--	--	-134 2 500 s	2 62 s	--	-206 5.5 s	--	--	--	--	--	--	--	--	--	0 0.61 s	**48** **400 s**	**2** **6.4 s**	**2** **5.9 s**	--
BitVectors 47 tasks, max. score: 83	--	4 58 s	52 16 000 s	**68** **1 800 s**	**58** **870 s**	--	**69** **470 s**	--	--	--	--	--	--	--	--	--	-80 550 s	--	5 170 s	-62 120 s	--
Concurrency 1 003 tasks, max. score: 1 222	--	--	--	**1 039** **78 000 s**	0 0 s	--	1 014 13 000 s	--	--	--	--	**1 222** **5 600 s**	--	**1 222** **16 000 s**	--	--	- 8 973 42 s	--	--	--	984 36 000 s
ControlFlow 1 927 tasks, max. score: 3 122	--	--	537 43 000 s	158 570 000 s	**2 317** **47 000 s**	--	**1 968** **59 000 s**	--	--	--	--	--	--	--	--	--	**2 169** **30 000 s**	1 691 78 000 s	1 887 54 000 s	872 10 000 s	--
ControlFlowInteger 48 tasks, max. score: 78	--	--	38 11 000 s	62 1 300 s	77 1 800 s	--	78 87 s	--	--	--	--	--	--	--	--	--	77 440 s	61 200 s	78 990 s	43 1 000 s	--
ECA 1 140 tasks, max. score: 1 874	--	--	0 0 s	- 2 334 560 000 s	987 39 000 s	--	523 58 000 s	--	--	--	--	--	--	--	--	--	516 25 000 s	112 45 000 s	632 46 000 s	1 14 s	--
Loops 142 tasks, max. score: 235	--	--	46 32 000 s	59 4 900 s	118 2 600 s	--	66 620 s	168 45 000 s	--	--	--	--	--	--	125 1 600 s	--	130 1 100 s	86 370 s	115 2 900 s	109 4 300 s	--
ProductLines 597 tasks, max. score: 929	--	--	0 0 s	399 1 900 s	901 4 100 s	--	917 430 s	--	--	--	--	--	--	--	--	--	913 3 900 s	917 32 000 s	554 3 300 s	87 5 000 s	--
DeviceDrivers64 1 650 tasks, max. score: 3 097	--	--	--	2 293 380 000 s	**2 572** **39 000 s**	--	2 281 36 000 s	--	--	--	--	--	--	--	--	--	**2 657** **16 000 s**	2 507 72 000 s	274 850 s	82 270 s	--
Floats 81 tasks, max. score: 140	--	--	--	**129** **15 000 s**	**78** **5 100 s**	--	**-12** **5 300 s**	--	--	--	--	--	--	--	--	--	-164 5.9 s	--	--	--	--
HeapManipulation 80 tasks, max. score: 135	--	--	70 6 000 s	**100** **13 000 s**	96 930 s	--	79 37 s	--	32 1.8 s	--	--	--	--	--	--	**111** **140 s**	-37 14 s	**109** **820 s**	84 460 s	84 420 s	--
MemorySafety 205 tasks, max. score: 361	--	--	**200** **82 000 s**	-433 14 000 s	**326** **5 700 s**	--	--	--	22 25 s	--	--	--	28 2 100 s	--	--	**221** **460 s**	0 0 s	--	95 13 000 s	66 4 800 s	--
Recursive 24 tasks, max. score: 40	--	6 22 s	--	0 10 000 s	16 31 s	**18** **140 s**	--	--	--	--	--	--	--	--	--	--	-88 2.3 s	**27** **2 300 s**	**25** **310**	10 220	--
Sequentialized 261 tasks, max. score: 364	--	--	--	-171 39 000 s	**130** **11 000 s**	--	**193** **9 600 s**	--	--	--	--	--	--	--	--	--	-59 5 800 s	--	**15** **8 600 s**	-10 7 000 s	--
Simple 46 tasks, max. score: 68	--	--	--	51 16 000 s	**54** **4 000 s**	--	29 990 s	--	--	--	--	--	--	--	--	--	**65** **1 400 s**	**51** **5 100 s**	0 1 800 s	3 140 s	--
Termination 393 tasks, max. score: 742	**610** **5 400 s**	--	--	--	0 0 s	--	--	--	--	350 61 s	**545** **300 s**	--	--	--	--	--	0 0 s	--	**565** **8 600 s**	--	--
Overall 5 803 tasks, max. score: 9 562	--	--	--	**1 731** **1 100 000 s**	**4 889** **110 000 s**	--	- 2 161 130 000 s	--	--	--	--	--	--	--	--	--	- 6 228 53 000 s	--	**2 301** **87 000 s**	231 23 000 s	--

John
Software Engineer

Competition on Software Verification (SV-COMP) 2015

John
Software Engineer

Verify my safety-critical code!

tools

competition categories

bounded model checking

predicate abstraction

concurrency support?

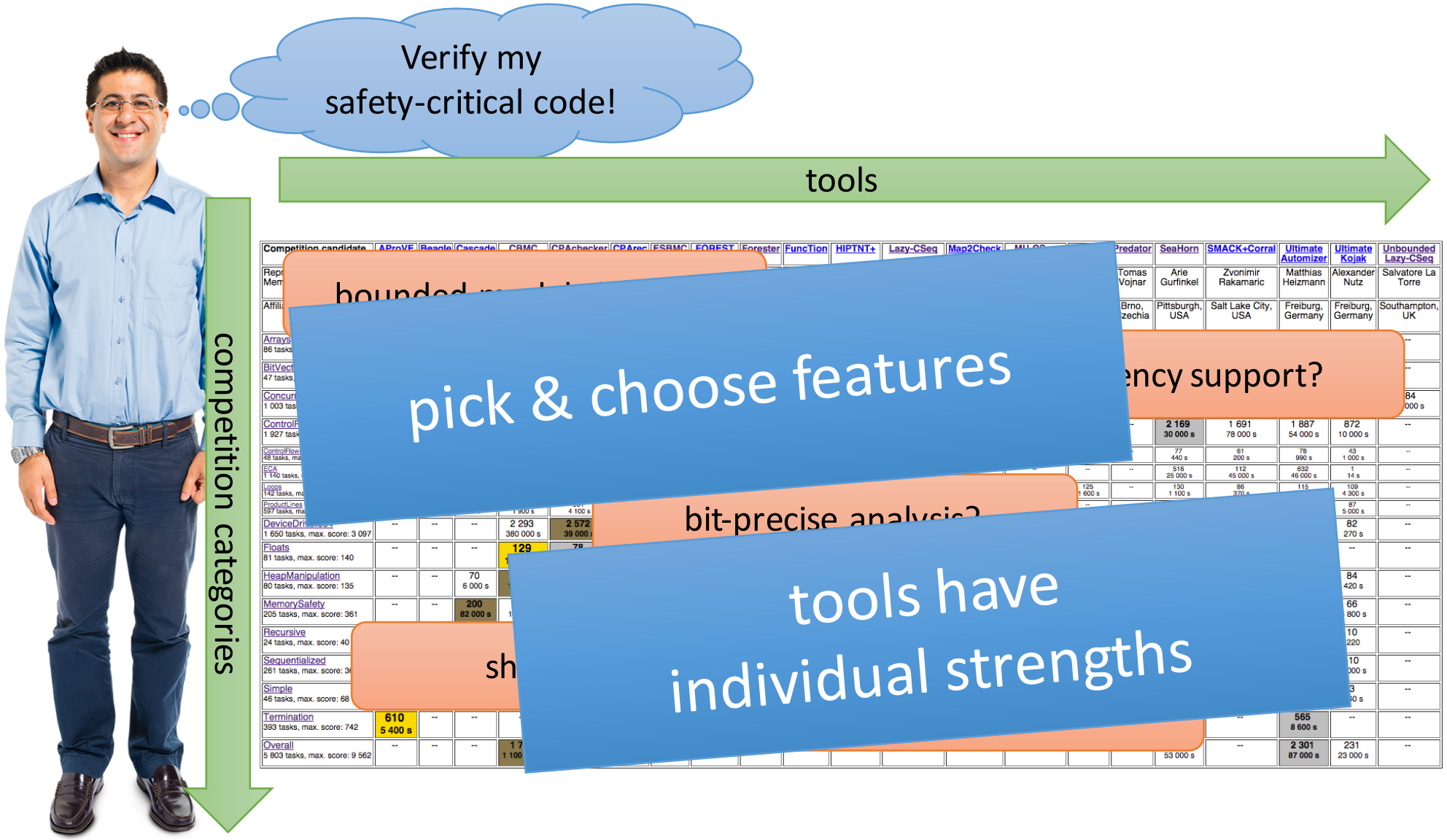
bit-precise analysis?

shape analysis

separation logic

Competition candidate	AProVE	Beagle	Cascade	CBMC	CPAchecker	CPArec	ESBMC	FOREST	Forester	FuncTion	HIPTNT+	Lazy-CSeq	Map2Check	MU-CSeq	Perentie	Predator	SeaHorn	SMACK+Corral	Ultimate Automizer	Ultimate Kojak	Unbounded Lazy-CSeq
Repr / Mem										Caterina Urban	Ton-Chanh Le	Gennaro Parlato	Herbert Oliveira Rocha	Bernd Fischer	Franck Cassez	Tomas Vojnar	Arie Gurfinkel	Zvonimir Rakamaric	Matthias Heizmann	Alexander Nutz	Salvatore La Torre
Affili										Paris, France	Singapore, Singapore	Southampton, UK	Manaus, Brazil	Stellenbosch, South Africa	Sydney, Australia	Brno, Czechia	Pittsburgh, USA	Salt Lake City, USA	Freiburg, Germany	Freiburg, Germany	Southampton, UK
Arrays 86 tasks, max. sc											--	--									--
BitVectors 47 tasks, max. sc											--	--									--
Concurrency 1 003 tasks, max.											--	1 22 5 600									84 000 s
ControlFlow 1 927 tasks, max. score: 3 122	--	--	537 43 000 s	158 570 000 s	**2 317** **47 000 s**	--	1 968 59 000 s	--	--	--	--	--	--	--	--	--	**2 169** **30 000 s**	1 691 78 000 s	1 887 54 000 s	872 10 000 s	--
ControlFlowInteger 48 tasks, max. score: 78	--	--	38 11 000 s	62 1 300 s	77 1 800 s	--	78 87 s	--	--	--	--	--	--	--	--	--	77 440 s	61 200 s	78 990 s	43 1 000 s	--
ECA 1 140 tasks, max. score: 1 874	--	--	0 0 s	- 2 334 560 000 s	987 39 000 s	--	523	--	--	--	--	--	--	--	--	--	516 25 000 s	112 45 000 s	632 46 000 s	1 14 s	--
Loops 142 tasks, max. score: 235	--	--	46 32 000 s	59 4 900 s	118 2 600 s										125 600 s	--	130 1 100 s	86 370 s	115 2 900 s	109 4 300 s	--
ProductLines 597 tasks, max. score: 929	--	--	0 0 s	399 1 900 s	901 4 100 s										--	--	913 3 900 s	917 32 000 s	554 3 300 s	87 5 000 s	--
DeviceDrivers64 1 650 tasks, max. score: 3 097	--	--	--	2 293 380 000 s	**2 572** **39 000**										--	--	**2 657** **16 000 s**	2 507 72 000 s	274 850 s	82 270 s	--
Floats 81 tasks, max. score: 140	--	--	--	**129** **15 000 s**	**78** **5 100 s**										--	--	-164 5.9 s	--	--	--	--
HeapManipulation 80 tasks, max. score: 135	--	--	70 6 000 s	**100** **13 000 s**	96 930 s	--	79 37 s	--	32 1.8 s	--	--	--	--	--	--	**111** **140 s**	-37 14 s	**109** **820 s**	84 460 s	84 420 s	--
MemorySafety 205 tasks, max. score: 361	--	--	**200** **82 000 s**	-433 14 000 s	**326** **5 700 s**	--	--	--	22 25 s	--	--	--	28 2 100 s	--	--	**221** **460 s**	0 0 s	--	95 13 000 s	66 4 800 s	--
Recursive 24 tasks, max. score: 40												--	--	--	--	--	-88 2.3 s	**27** **2 300 s**	**25** **310**	10 220	--
Sequentialized 261 tasks, max. score: 3												--	--	--	--	--	-59	--	**15** **8 600 s**	-10 7 000 s	--
Simple 46 tasks, max. score: 68																		**51** **5 100 s**	0 1 800 s	3 140 s	--
Termination 393 tasks, max. score: 742	**610** **5 400 s**	--	--	--	0 0 s	--	--	--										--	**565** **8 600 s**	--	--
Overall 5 803 tasks, max. score: 9 562	--	--	--	1 731 1 100 000 s	**4 889** **110 000 s**	--	- 2 161 130 000 s	--									53 000 s	--	**2 301** **87 000 s**	231 23 000 s	--

Competition on Software Verification (SV-COMP) 2015



Helping John: Portfolio Solvers

Input

Property φ

Output

best tool
1. correct
2. minimal time

?

$tool_1$: Blast

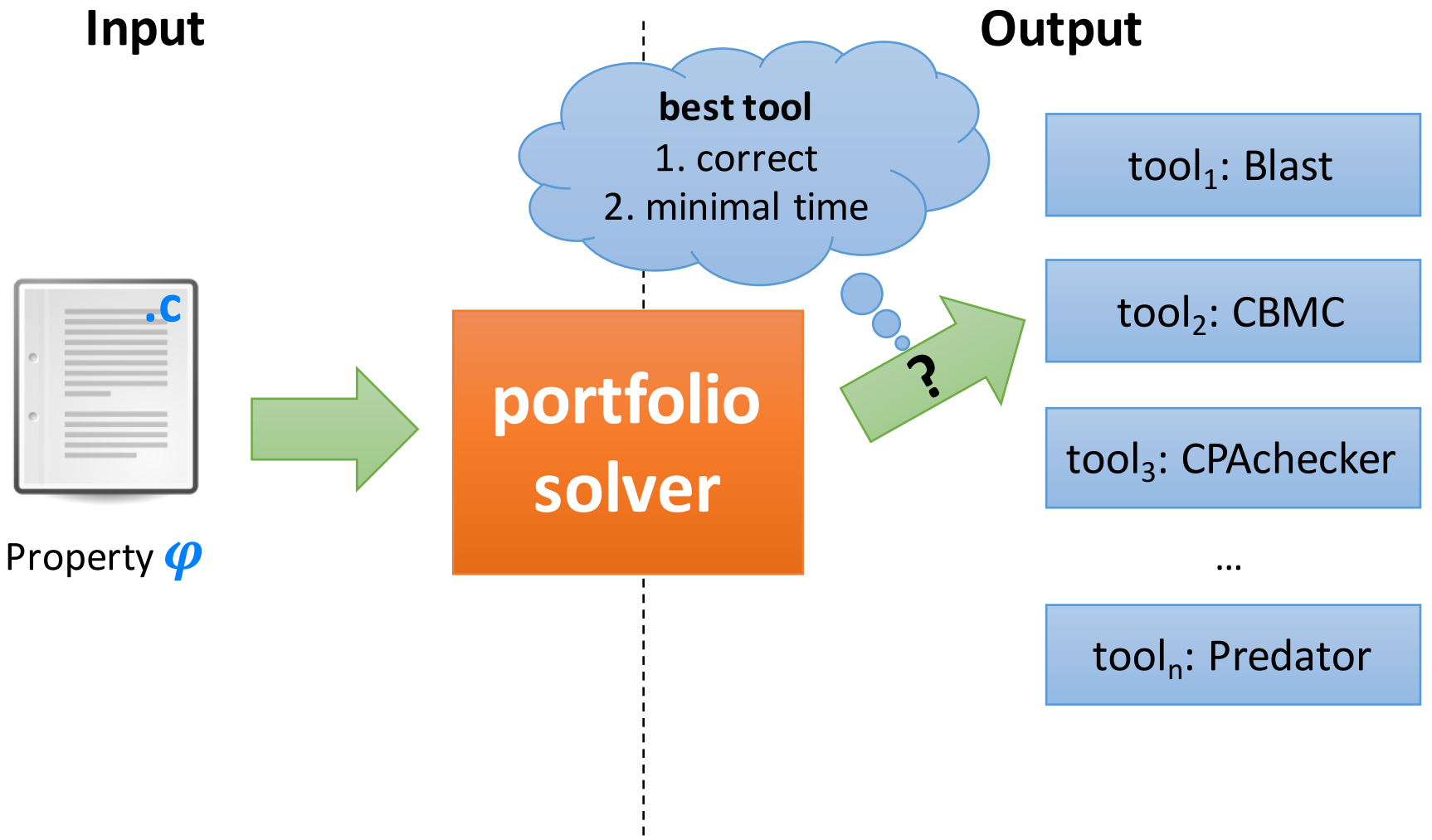
$tool_2$: CBMC

$tool_3$: CPAchecker

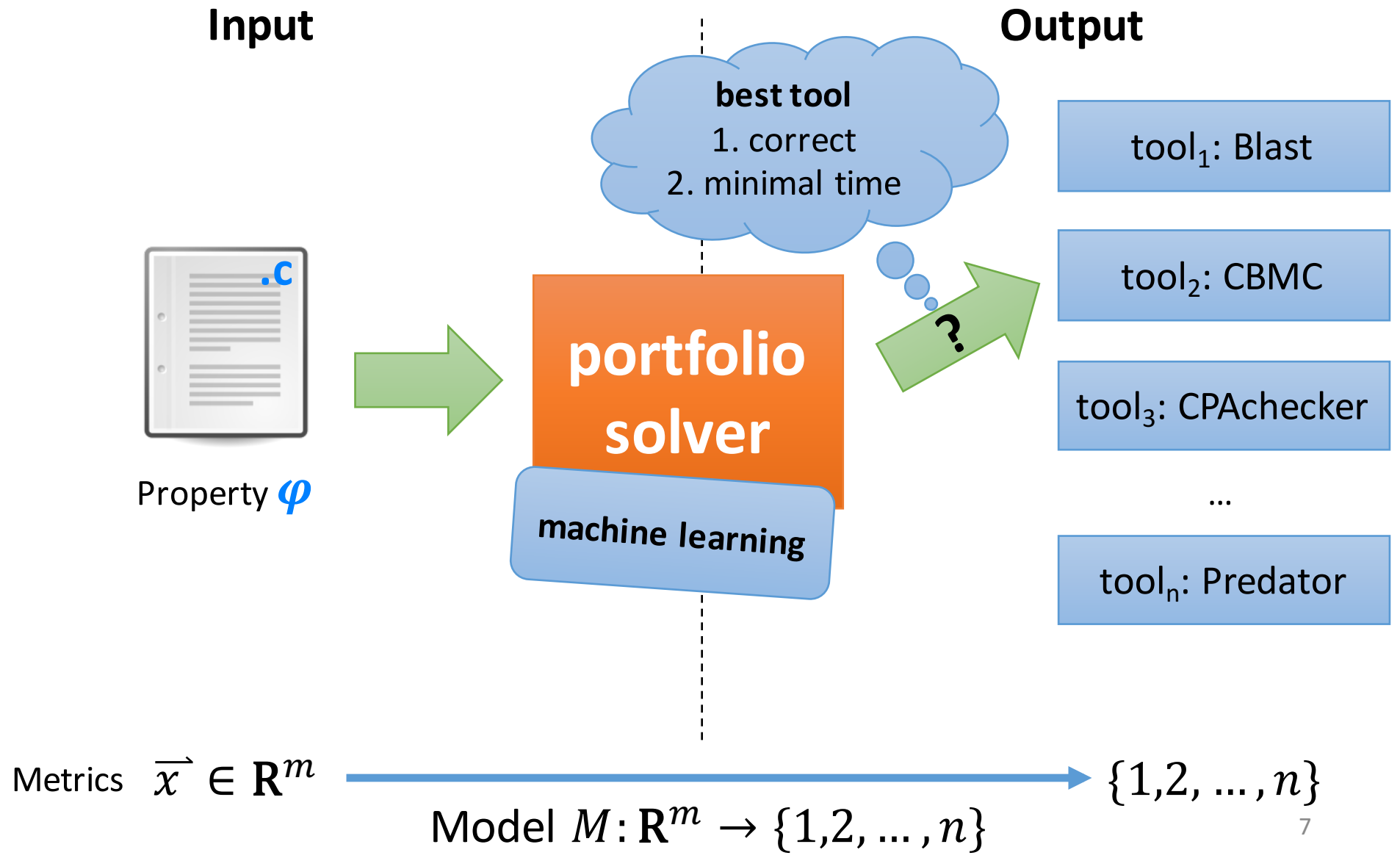
...

$tool_n$: Predator

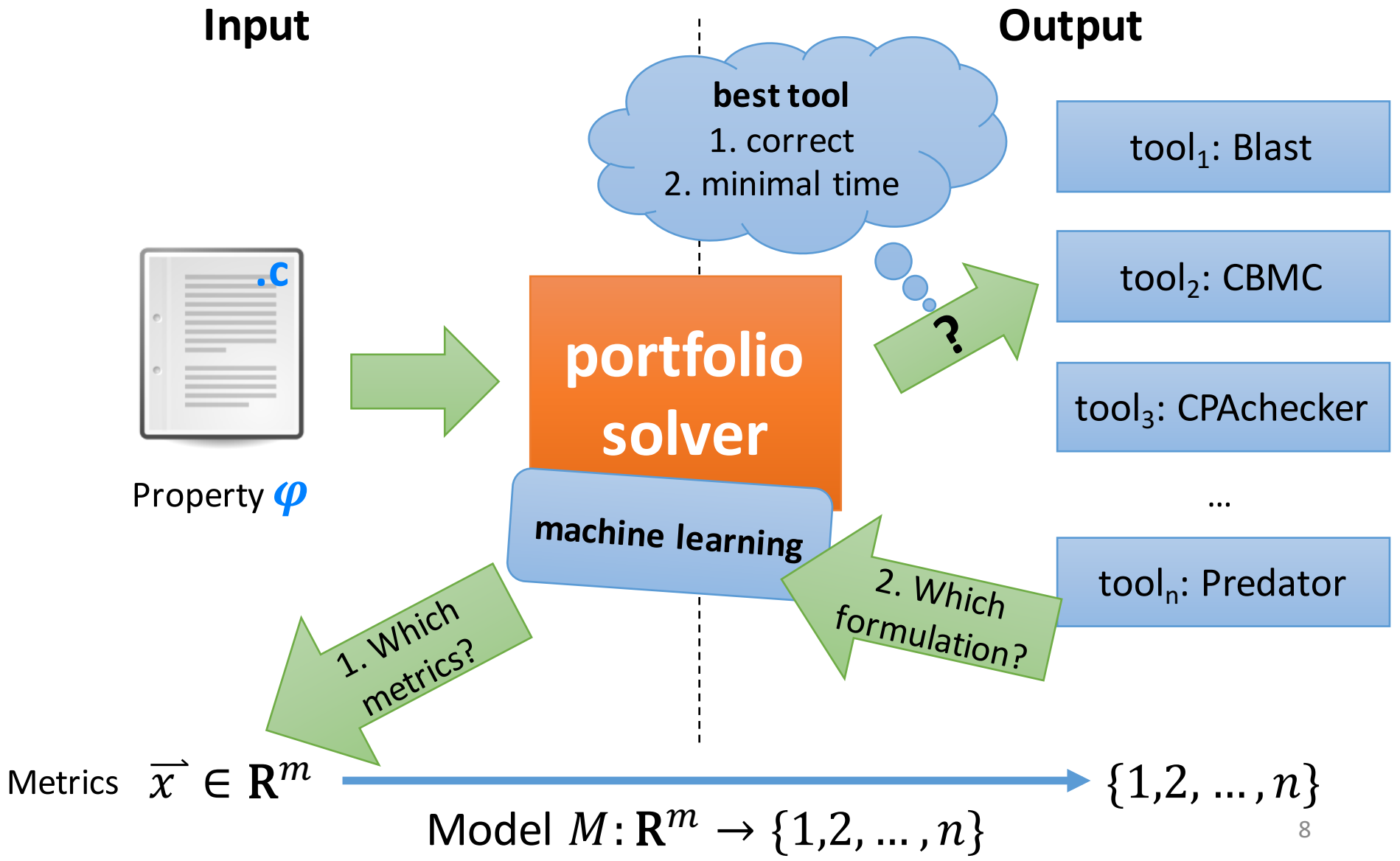
Helping John: Portfolio Solvers



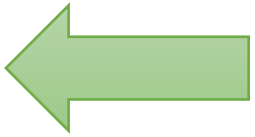
Helping John: Portfolio Solvers



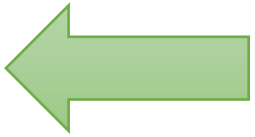
Helping John: Portfolio Solvers



Program Features – 3 Dimensions

Program Variables 

focus of this talk

Program Loops 

Control Flow

- Recursion?
- Function pointers?
- Complexity of the AST (# of nodes, # of branches)?

Variable Roles

Loop Patterns

Machine Learning

Variable Roles

Intuitively:
Usage patterns of variables by programmers.

Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
   n++;
   x = x & (x-1);
}
```

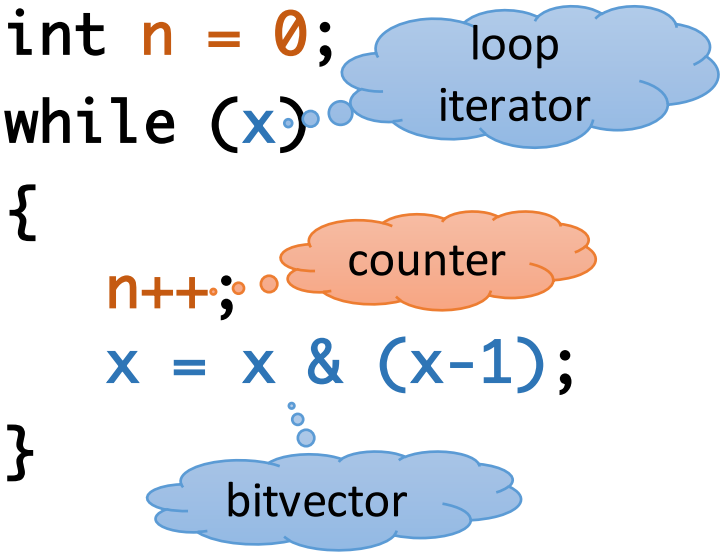
Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
    n++;
    x = x & (x-1);
}
```

Variable Roles

Intuitively:
Usage patterns of variables by programmers.



Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
    n++;
    x = x & (x-1);
}
```

loop iterator

counter

bitvector

```
int fd = open(path, flags);
int c, val = 0;
while (read(fd, &c, 1) > 0
       && isdigit(c))
{
    val = 10 * val + c - '0';
}
```

Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
    n++;
    x = x & (x-1);
}
```

loop iterator

counter

bitvector

```
int fd = open(path, flags);
int c, val = 0;
while (read(fd, &c, 1) > 0
       && isdigit(c))
{
    val = 10 * val + c - '0';
}
```

file descriptor

Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
    n++;
    x = x & (x-1);
}
```

loop iterator

counter

bitvector

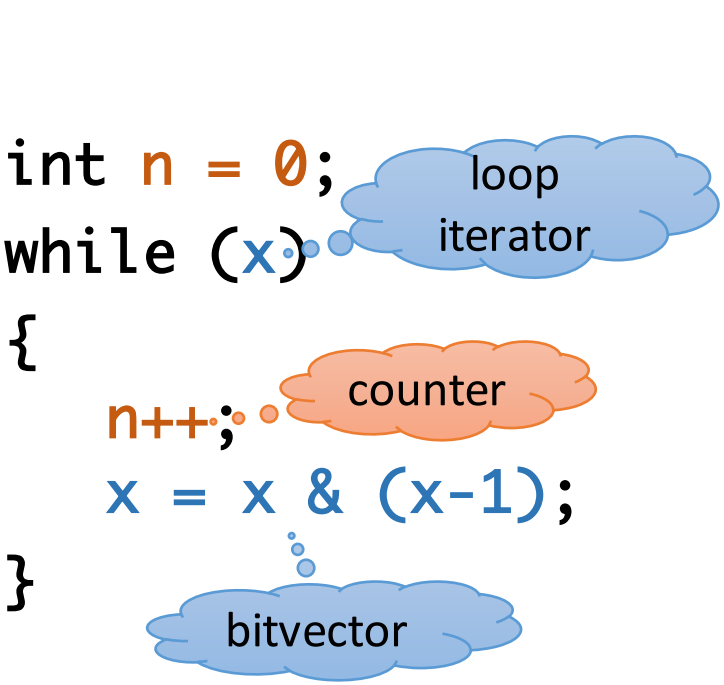
```
int fd = open(path, flags);
int c, val = 0;
while (read(fd, &c, 1) > 0
       && isdigit(c))
{
    val = 10 * val + c - '0';
}
```

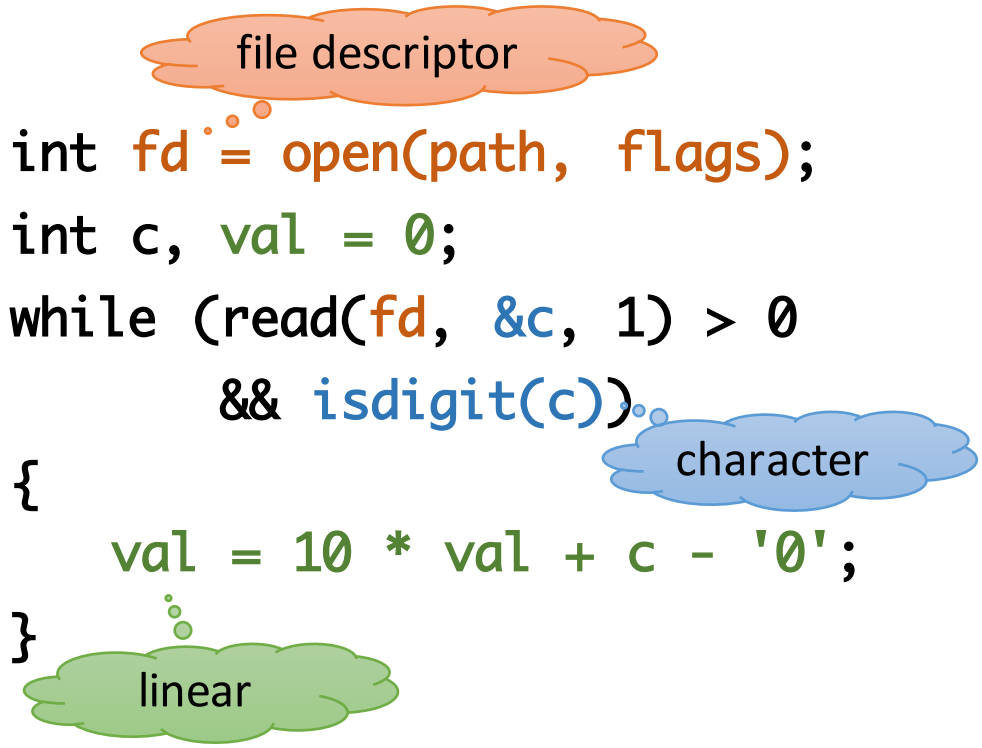
file descriptor

linear

Variable Roles

Intuitively:
Usage patterns of variables by programmers.





Variable Roles

Intuitively:
Usage patterns of variables by programmers.

```
int n = 0;
while (x)
{
    n++;
    x = x & (x-1);
}
```

loop iterator

counter

bitvector

```
int fd = open(path, flags);
int c, val = 0;
while (read(fd, &c, 1) > 0
       && isdigit(c))
{
    val = 10 * val + c - '0';
}
```

file descriptor

character

linear

distinct usage patterns

› abstraction functions?

yet, in C: all `int`

Variable Roles

27 variable roles

Choice

manual inspection of the cBench benchmark (5.2 KLOC)

Meaningfulness

1. widely used programming patterns
2. align with tool developer reports from SV-COMP
3. successfully used to predict SV-COMP category

FMCAD'13

Formal expression & computation

data flow analysis (control-flow insensitive, inter-procedural)

efficient!

Variable Roles · Metrics

$Vars$ = set of all variables

$Vars^R$ = set of variables assigned role *R*

$$m_R = \frac{|Vars^R|}{|Vars|}$$

Variable Roles

Loop Patterns

Machine Learning

Loops in SW Verification

Reasoning about loops is useful

- (un)reachability of assertions
- loop unrolling factors
- soundness limits for BMC
- …

But: Reasoning about loops is hard.

Hypothesis: reasoning about loops ~ termination analysis

There are loops for which termination proofs are easy.

How can we efficiently describe these loops?

Loop Patterns

Syntactically Terminating Loops

```
i=0;
while (i < N) {
   if (?) i+=1;
   else   i+=2;
}
```

$P(i) : i \geq N$

linear (in)equalities

constant updates

consistency constraints

Consistency constraints

- *P* only depends on *i*
- Updates to *i* entail eventual truth of *P*
- Control-flow reaches *P* when true

Termination Heuristics

```
i=0;
while (i < N) {
   if (?) i+=1;
   else   i+=2;
}
```

$P(i) : i \geq N$

linear (in)equalities

constant updates

consistency constraints

Consistency constraints

- *P* only depends on *i*
- Updates to *i* entail eventual truth of *P*
- Control-flow reaches

weaker

weaker

weaker

[Pani, 2014]: good heuristics for termination

Loop Patterns

5 loop patterns

Choice

pattern-based termination proof

derive termination heuristics by weakening constraints

Meaningfulness

1. widely used programming patterns
2. indicates hardness in comparison to termination proving tool

Pani'14

Formal expression & computation

data flow analysis (control-flow insensitive)

Loop Patterns · Metrics

$Loops =$ set of all loops

$L^C =$ set of loops in class C

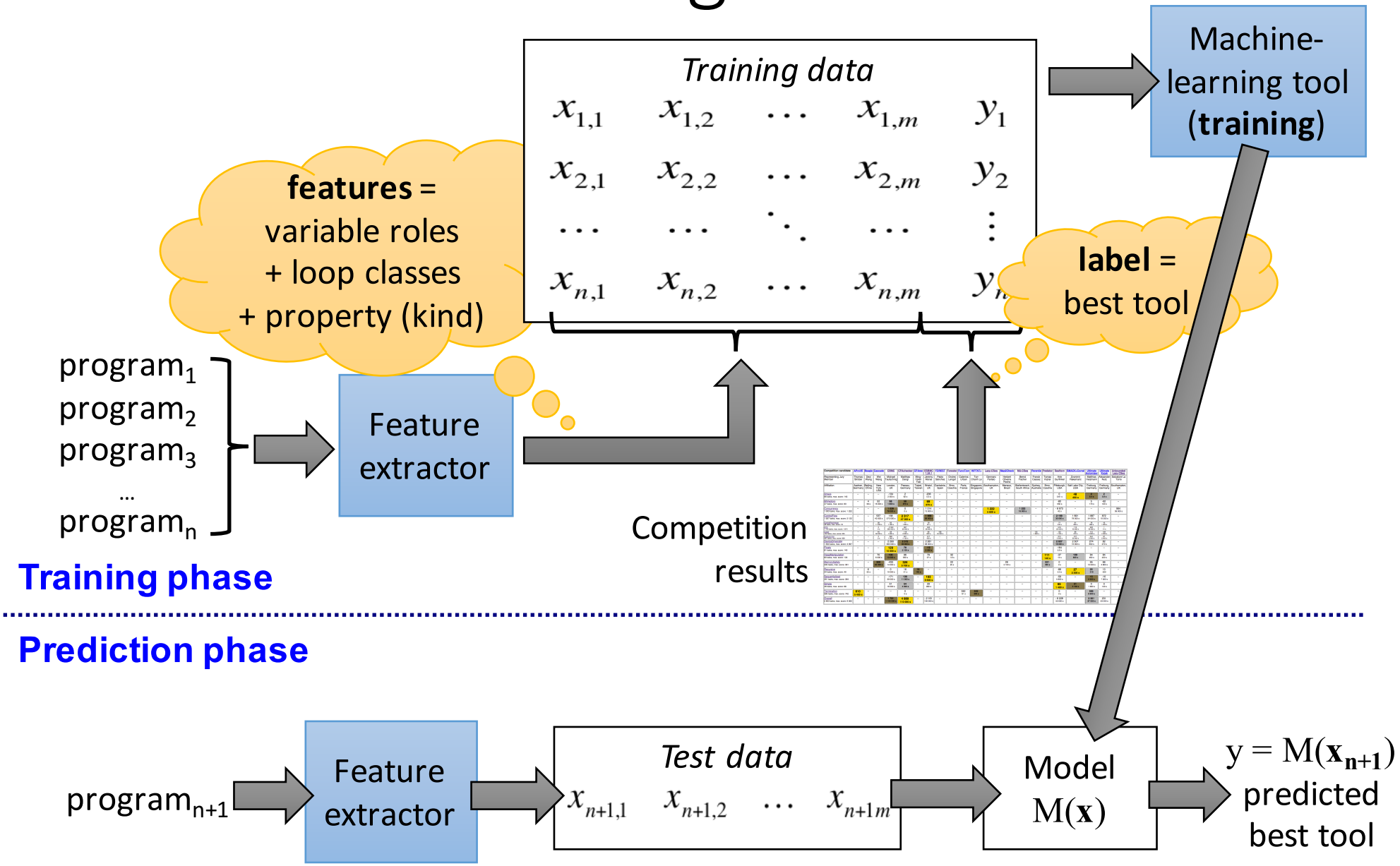
$$m_C = \frac{|L^C|}{|Loops|}$$

Variable Roles

Loop Patterns

Machine Learning

Machine Learning



Experiments

Based on data from SV-COMP'14 and '15.

Randomly split dataset 60:40 (*training* : *test*).

Repeat **10 experiments**:

Train on *training*.

Evaluate on *test*.

Report **arithmetic mean**.

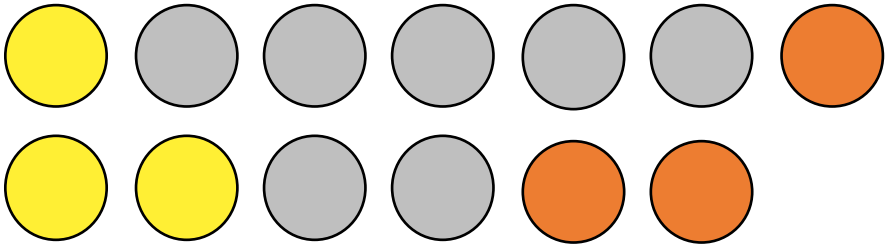
Experiments

heavily depends on scoring policy [CAV'15]

overall score | medals

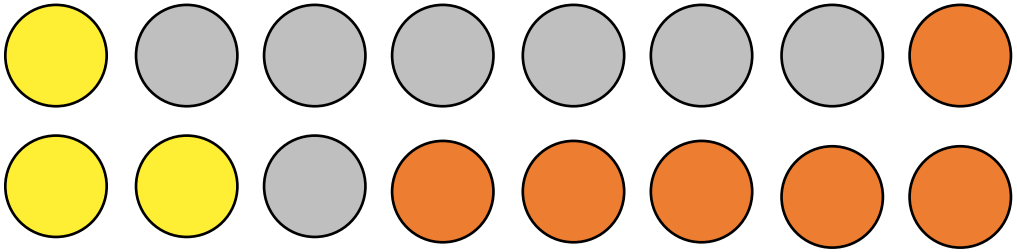
SV-COMP'14

Portfolio	1494
CBMC	1292

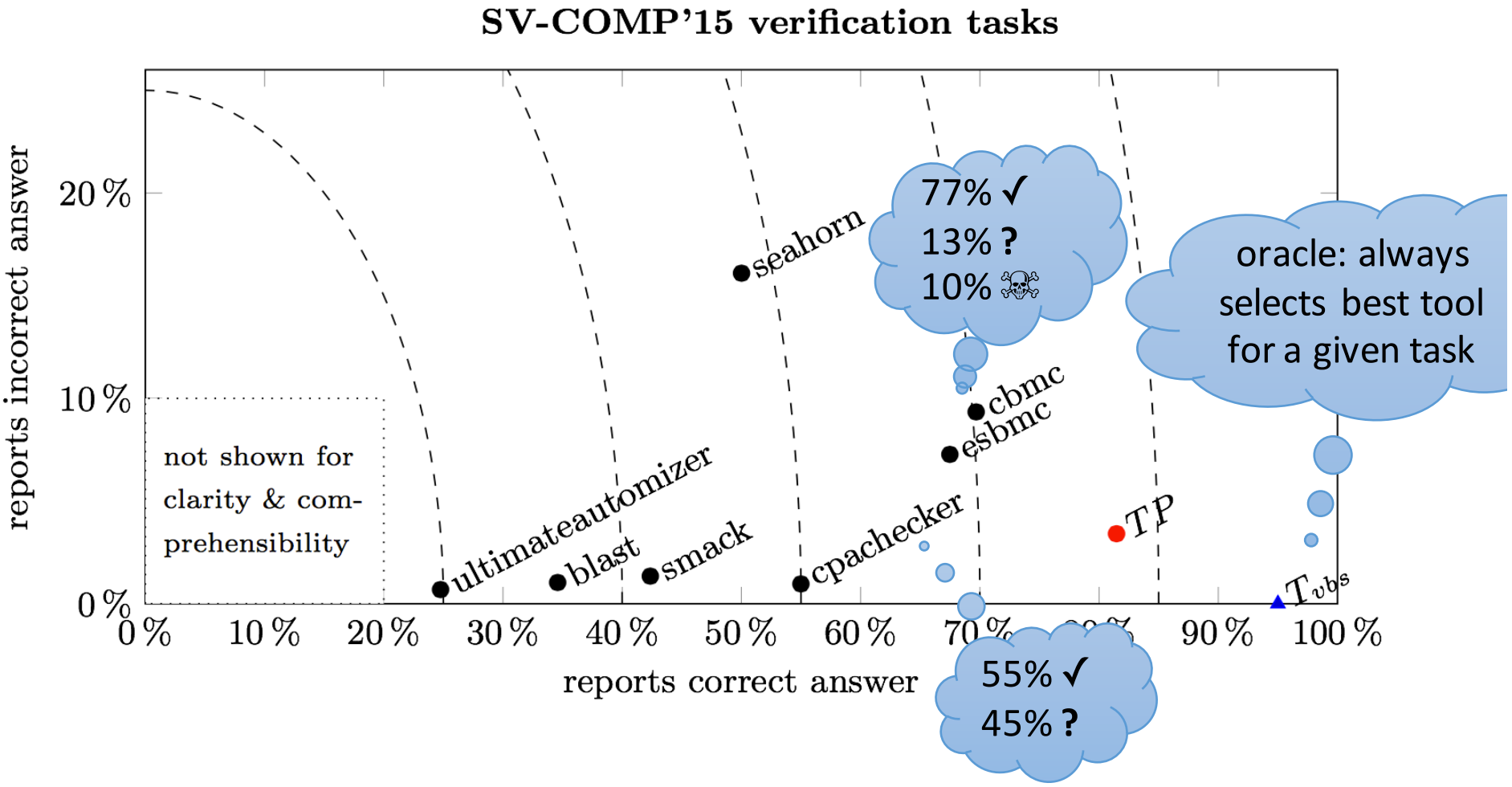


SV-COMP'15

Portfolio	2511
CPAchecker	2228



Experiments. Interpretation.



Conclusion

Introduce **2 families of program metrics**

- variable roles
- loop patterns

Build a **portfolio for SW verification**

- learns the best-performing tool from a training set
- metrics describe the benchmarks
- shows predictive power of our features

Future

- guide internal heuristics
- systematic description of benchmarks in research papers

Conclusion

Demyanova, Pani, Veith, Zuleger
Empirical Software Metrics for Benchmarking of Verification Tools
To appear in CAV'15.

» http://forsyte.at/software/verifolio/

tool, detailed results (soon)

References

[CAV'15] **Empirical Software Metrics for Benchmarking of Verification Tools**
Yulia Demyanova, Thomas Pani, Helmut Veith, Florian Zuleger
Computer Aided Verification - 27th International Conference, CAV 2015, 2015.

[Pani'14] **Loop Patterns in C Programs**
Thomas Pani
January 2014, Master's thesis, Vienna University of Technology.

[FMCAD'13] **On the Concept of Variable Roles and its Use in Software Analysis**
Yulia Demyanova, Helmut Veith, Florian Zuleger
FMCAD, pages 226-229, 2013.