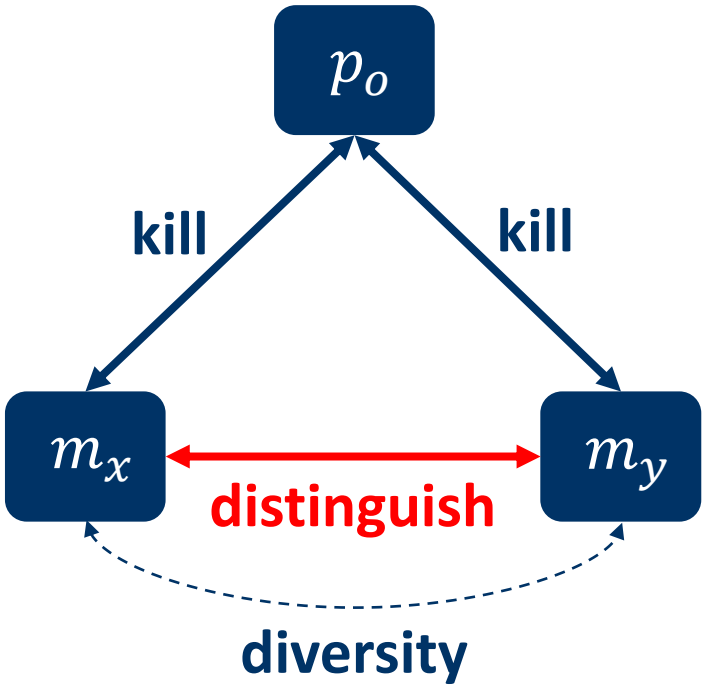
Diversity-Aware Mutation Adequacy Criterion for Improving Fault Detection Capability

Donghwan Shin, Shin Yoo, and Doo-Hwan Bae

KAIST, South Korea

@ Mutation 2016



What do you think about this?

- Assume: we have two mutants, and one test kills both.

1 = killed	m_x	m_y
t	1	1

✓ Happy because it forms a mutation-adequate test suite.

✓ Remove one of the mutants because they are redundant.

✓ Consider more tests to discover the diversity of the mutants.

It increases the fault detection capability with little effort!

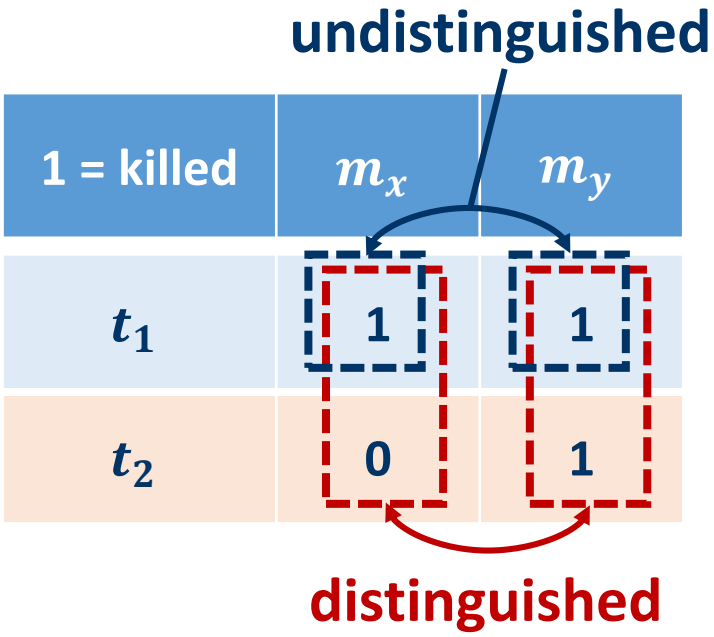
The concept of diversity has received much attention in software testing

- Adaptive Random Testing
- Clustering-based test selection and prioritization
- Information theoretic measures of diversity for test set selection

Diversity has received little attention in mutation

Considering the diversity of mutants for a set of tests

- Idea: *mutant distinguishment*
  - Two mutants are ***distinguished*** from each other by the set of tests that kill them (i.e., using kill patterns).
  - No extra information is required.



Overview of empirical evaluation results

* k-criterion: *kill-only* mutation adequacy criterion
d-criterion: *diversity-aware* mutation adequacy criterion

76.8 The amount of the most increased fault detection rate (pp) by the d-criterion in compared to the k-criterion.

Zero There is no fault whose fault detection rate of the d-criterion is statistically decreased in compared to the k-criterion.

1.67 The average size ratio of the test suites adequate to the d-criterion of the test suites adequate to the k-criterion.

Outline

Formalize the basic concepts: Kill and distinguish

Test differentiator*: Simple notation for the notion of difference

$$d(t, p_x, p_y) = \begin{cases} 1\ (true) & \text{, if the behaviors of } p_x \text{ and } p_y \text{ for } t \text{ are different} \\ 0\ (false) & \text{, otherwise} \end{cases}$$

Concepts	Notations
"A test *kills* a mutant."	$d(t, p_o, m) = 1$
"A test suite kills all mutants."	$\forall m \in M, \exists t \in TS, d(t, p_o, m) = 1$ = Existing mutation adequacy criterion (k-criterion)

* D. Shin and D. H. Bae, "A theoretical framework for understanding mutation-based testing methods," in Proceedings of the IEEE International Conference on Software Testing, Verification and Validation (ICST 2016). IEEE, to appear.

Mutant distinguishment: Two mutants are distinguished from each other by a test

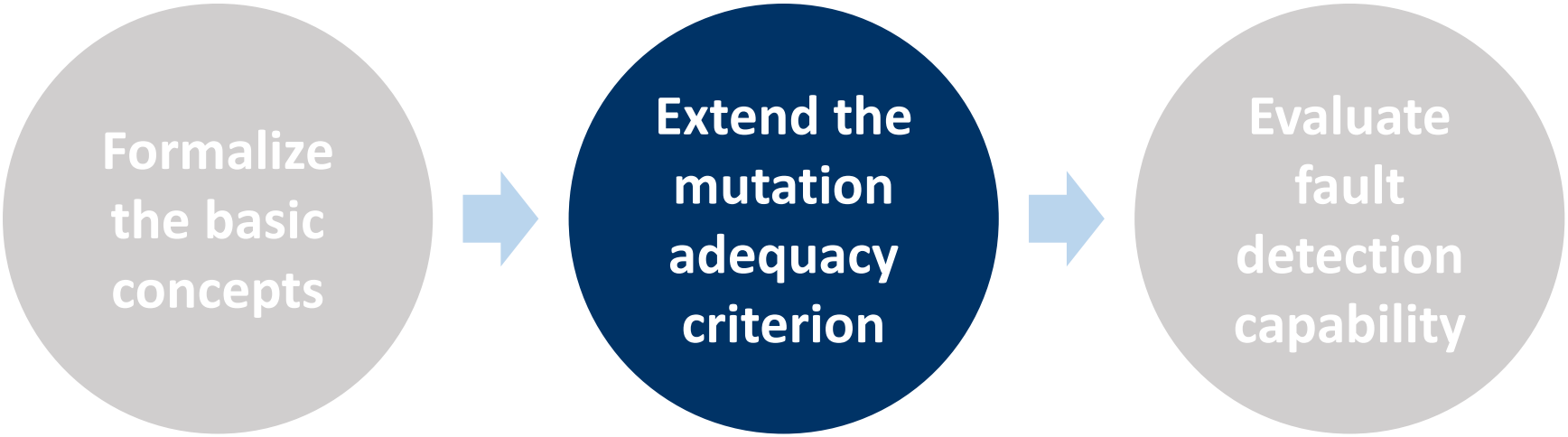
A test t ***distinguishes*** two mutants m_x and m_y:

$$d(t, p_o, m_x) \neq d(t, p_o, m_y)$$

A set of tests TS ***distinguishes*** two mutants m_x and m_y:

$$\exists t \in TS, d(t, p_o, m_x) \neq d(t, p_o, m_y)$$

Extend the mutation adequacy criterion from “kill-only” to “diversity-aware”



Distinguishing mutation adequacy criterion: extended mutation adequacy considering diversity of mutants

A set of tests TS is *distinguishing mutation-adequate:*

$$\forall m_x, m_y \in M', \exists t \in TS, d(t, p_o, m_x) \neq d(t, p_o, m_y)$$

where $m_x \neq m_y$, $M' = M \cup \{m_o\}$, and $m_o = p_o$

- **Interesting properties**
  - Formally subsumes the k-criterion (thanks to M').
  - Need to compare only at most $|M|$ mutants, not $\binom{|M|}{2} \approx |M|^2$.

Working example: achieving d-adequate test suites

$d(t_j, p_o, m_i)$	m_1	m_2	m_3	m_4
t_1	1	1	0	0
t_2	0	0	1	1
t_3	1	0	1	0

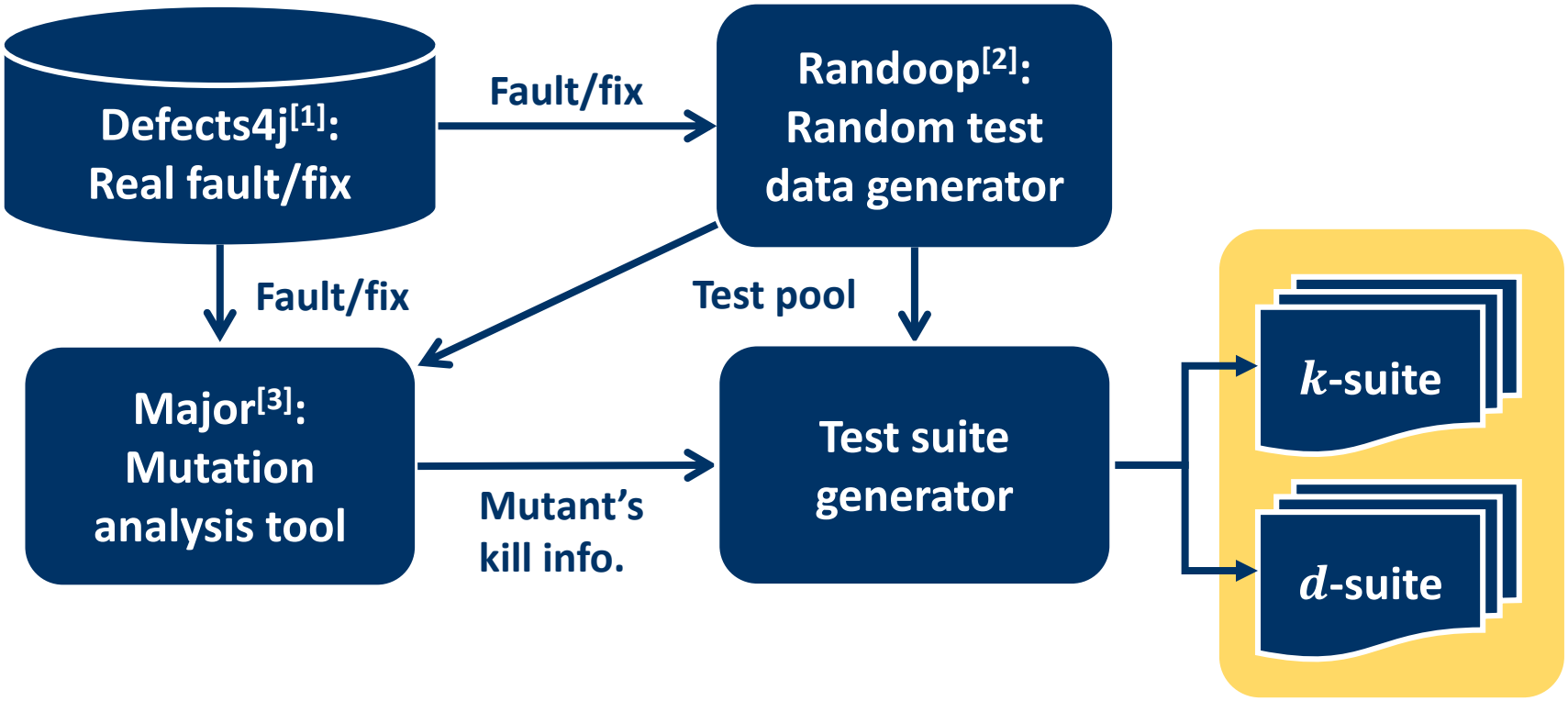
Test suite	Distinguished mutant sets	Number of mutant pairs to be compared
$\{\}$	$\{m_o, m_1, m_2, m_3, m_4\}$	4 (0-1,0-2,0-3,0-4)
$\{t_1\}$	$\{m_o, m_3, m_4\}, \{m_1, m_2\}$	3 (0-3,0-4,1-2)
$\{t_1, t_2\}$	$\{m_o\}, \{m_1, m_2\}, \{m_3, m_4\}$	2 (1-2, 3-4)
$\{t_1, t_2, t_3\}$	$\{m_o\}, \{m_1\}, \{m_2\}, \{m_3\}, \{m_4\}$	0

Evaluate the fault detection capabilities of mutation adequacy criteria

Research questions

- RQ1: Are there rooms for improvement of the fault detection capability by distinguishing more mutants?
- RQ2: Is the d-criterion more likely to detect faults than the k-criterion?
- RQ3: How many tests are needed to be a mutation- adequate test suite?
- (RQ4: How much time takes for selecting a mutation- adequate test suite?)

Evaluation setup



[1] Just, René, Darioush Jalali, and Michael D. Ernst. "Defects4J: A Database of existing faults to enable controlled testing studies for Java programs." Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014.

[2] Pacheco, Carlos, and Michael D. Ernst. "Randoop: feedback-directed random testing for Java." Companion to the 22nd ACM SIGPLAN conference on Object-oriented programming systems and applications companion. ACM, 2007.

[3] Just, René. "The Major mutation framework: Efficient and scalable mutation analysis for Java." Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014.

Fault detection capability of an adequacy criterion

For each fault,

Fault detection capability of the d-criterion (or k-criterion)

= Fault detection rate of the d-suite (or k-suite)

= % fault-detecting test suites in the d-suite (or k-suite)

= 500 independent d-criterion adequate test suites

= 500 independent k-criterion adequate test suites

Subject faults, tests, and mutants

Fault	Test pool size	Test pool trigs	Mutation analysis allM	Mutation analysis kM	Mutation analysis dM	Mutation analysis time (sec)	Fault	Test pool size	Test pool trigs	Mutation Analysis allM	Mutation Analysis kM	Mutation Analysis dM	Mutation Analysis time (sec)
Chart-5	10000	165	271	168	110	2157	Math-9	3788	1	84	51	31	106
Chart-11	236	34	221	26	13	82	Math-14	5492	2	24	10	6	206

- **Total 45 isolated real faults from five applications**
- **At most 10,000 tests for each fault as a random test pool**
- **Generate mutants using the five commonly used mutation operators[1][2]**

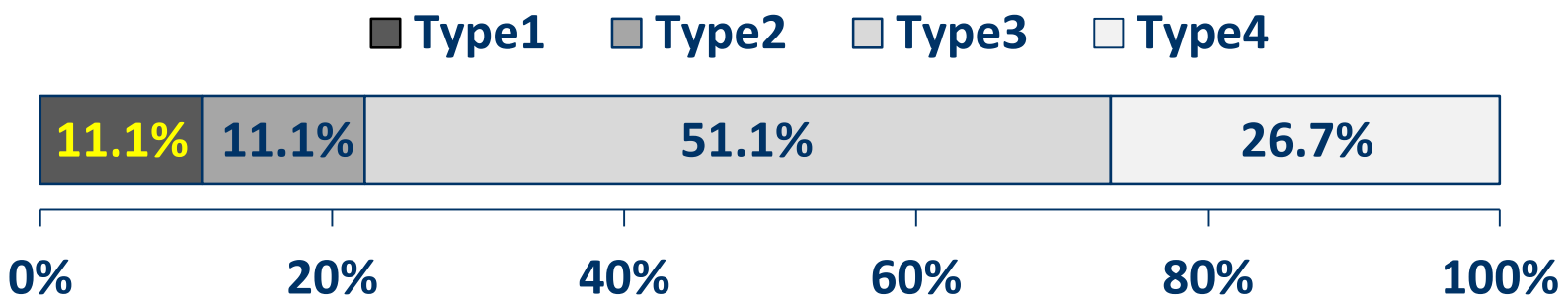
Lang-56	3707	381	495	337	246	12223	Math-102	2844	46	282	206	61	217
Lang-59	1199	28	1412	652	321	408	Math-103	5878	226	108	91	75	707

[1] A. Siami Namin, J. H. Andrews, and D. J. Murdoch. Sufficient mutation operators for measuring test effectiveness. In Proceedings of the International Conference on Software Engineering (ICSE), pages 351–360, 2008.

[2] A.J. Offutt, A. Lee, G. Rothermel, R.H. Untch, and C. Zapf, "An Experimental Determination of Sufficient Mutation Operators," ACM Trans. Software Eng. and Methodology, vol. 5, no. 2, pp. 99-118, 1996.

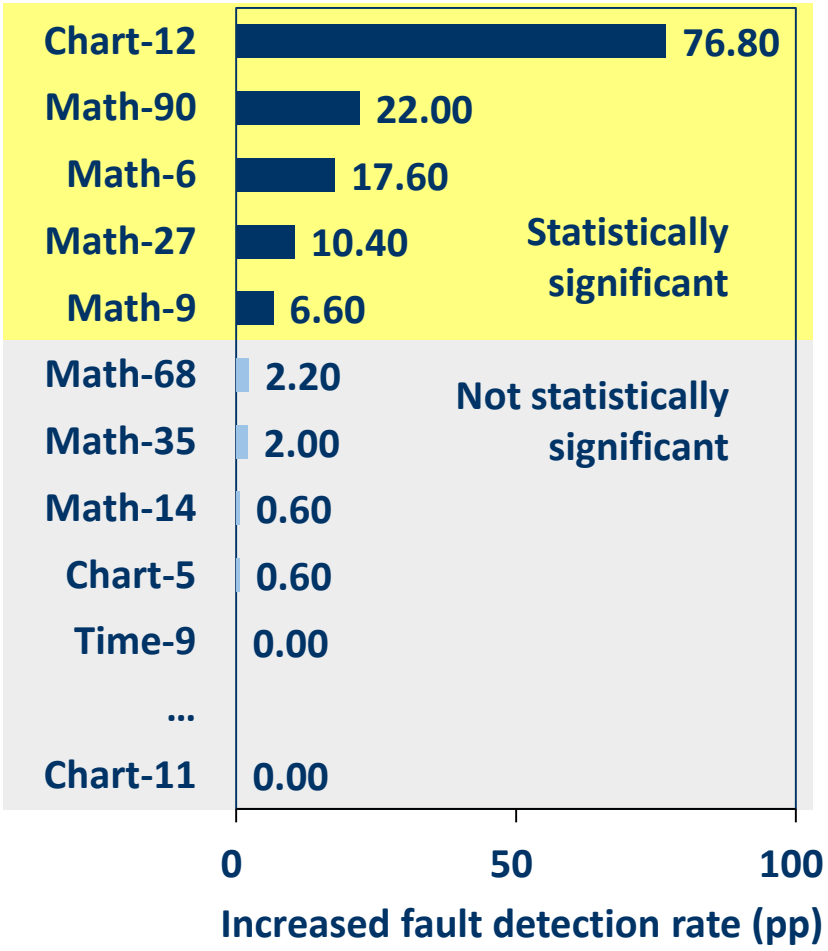
RQ1: Are there rooms for improvement of fault detection capability by distinguishing more mutants?

	Fault detection rate of k-suite < 1	Fault detection rate of k-suite = 1
Distinguished mutants rate of k-suite < 1	Type1: the room for improvement	Type3: the strong point of the k-criterion
Distinguished mutants rate of k-suite = 1	Type2: the weak point of the d-criterion	Type4: implicitly considered diversity of mutants



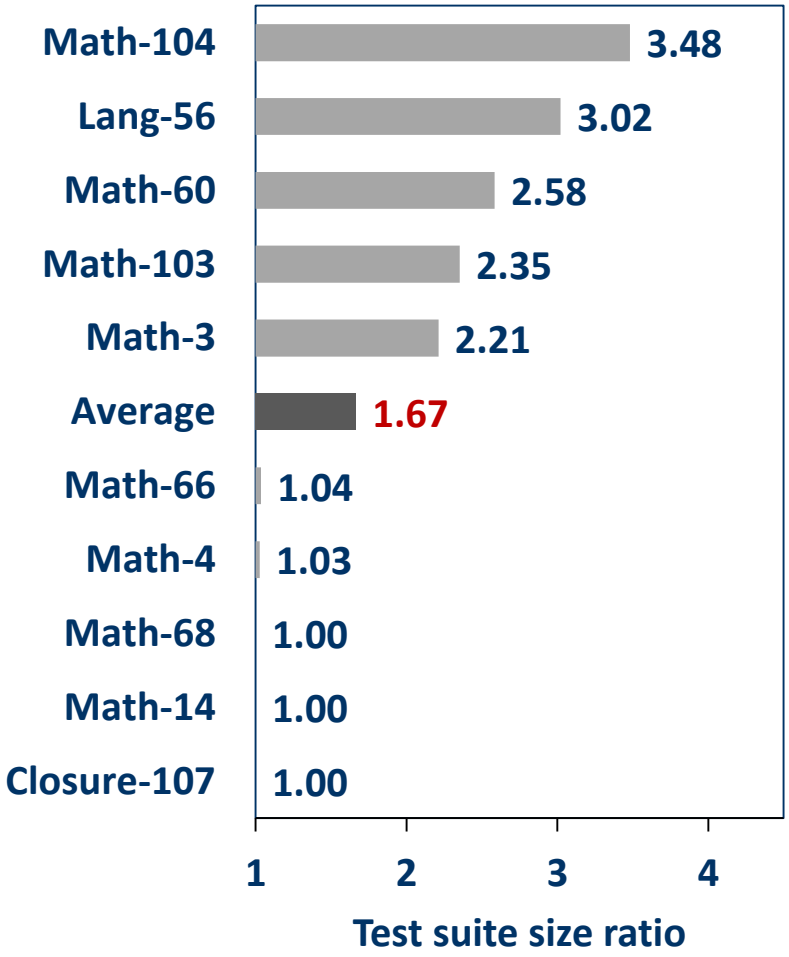
RQ2: Is the d-criterion more likely to detect faults than the k-criterion?

- **Yes, for all type1 faults!**
  - Statistically significant (α=0.05, non-parametric prop. test)
  - The effect size is also significant (increased at most 76.8 pp)
- **Never worse!**
  - The d-criterion is as effective as the k-criterion in the worst case.



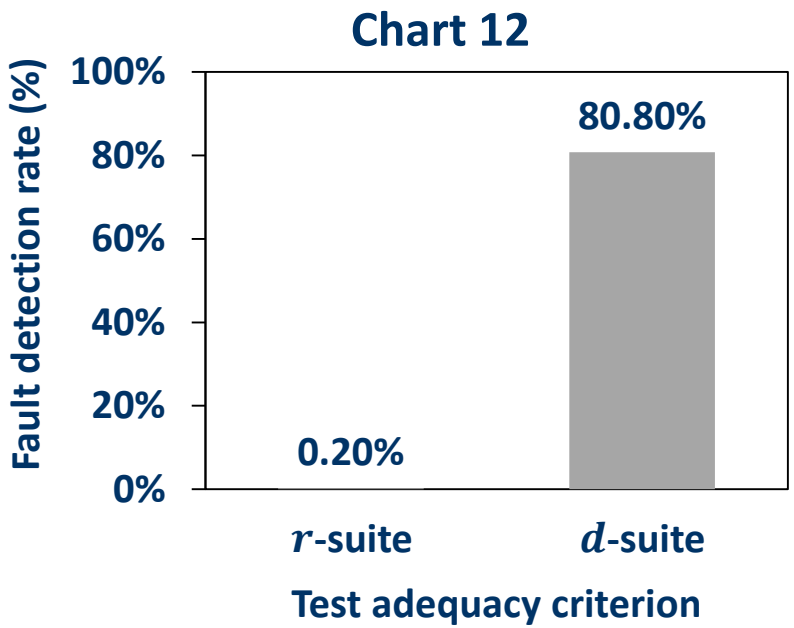
RQ3: How many tests are needed to be a mutation-adequate test suite?

- In average, the d-criterion needs **1.67 times more tests** than the k-criterion.

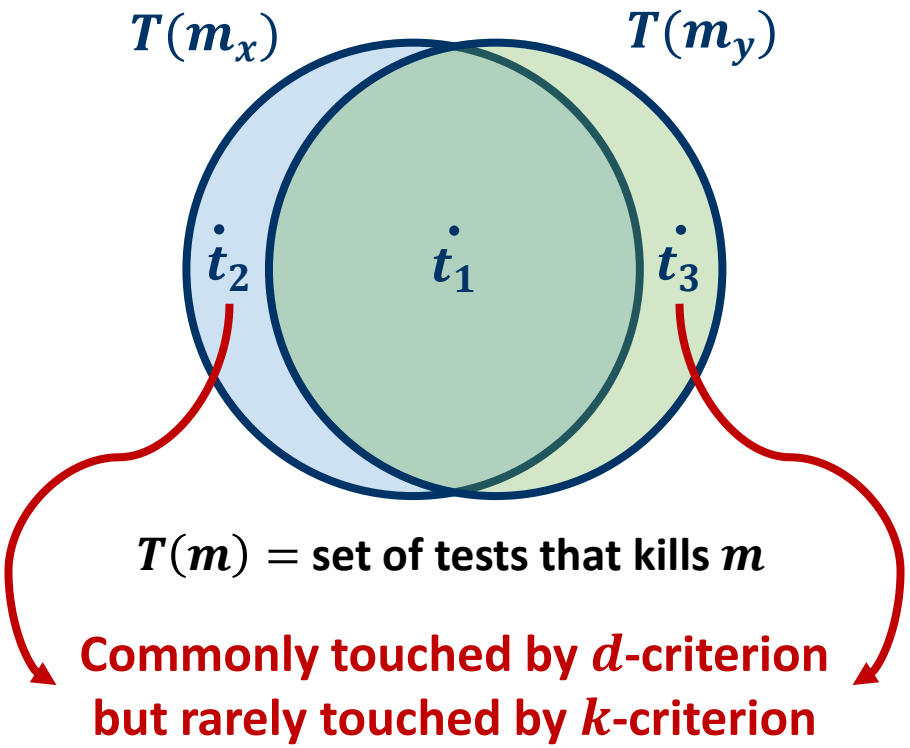


Closer look: why the d-criterion is superior than the k-criterion in some cases?

- Test suite size effect? **NO!**



- One possible explanation



Discussion: undistinguished or redundant?

- **Assume: we have only two mutants, and one test kills both.**

For the two mutants	"They are undistinguished"	"They are redundant"
Intention	Try to add more tests to distinguish mutants	Try to remove mutants to minimize the set of mutants
Basic idea	Distinguishing mutants improves the fault detection	Mutation score is inflated as a measure of the fault detection
Balance between TS and M	Improve TS for M	Reduce M for TS
Representative paper	This work	Ammann et al.* (ICST 2014)

* Ammann, Paul, Marcio E. Delamaro, and Jeff Offutt. "Establishing theoretical minimal sets of mutants." Software Testing, Verification and Validation (ICST), 2014 IEEE Seventh International Conference on. IEEE, 2014.

In summary, considering the diversity of mutants improves the fault detection capability

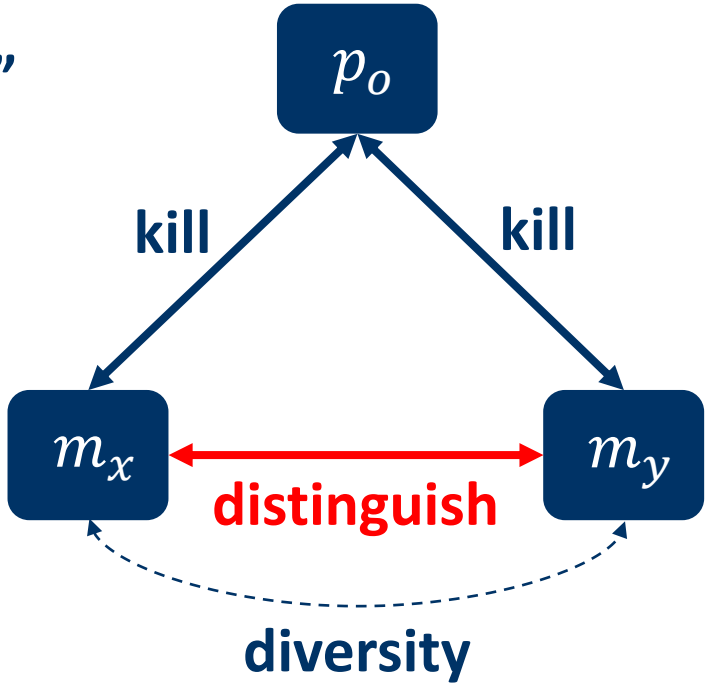
"All pair of mutants are *distinguished* by a test suite."

better than

"All mutants are *killed* by a test suite."

in terms of fault detection capability

Questions?



Appendix: d-criterion formally subsumes k-criterion

Start from the definition of d-criterion:

$$\forall m_x, m_y \in M', \exists t \in TS, d(t, p_o, m_x) \neq d(t, p_o, m_y)$$

(limit the scope of mutants) Let $m_y = p_o \in M'$:

$$\forall m_x, p_o \in M', \exists t \in TS, d(t, p_o, m_x) \neq d(t, p_o, p_o)$$

$$\forall m_x, p_o \in M', \exists t \in TS, d(t, p_o, m_x) \neq 0$$

$$\forall m_x, p_o \in M', \exists t \in TS, d(t, p_o, m_x) = 1$$

$$\forall m \in M, \exists t \in TS, d(t, p_o, m) = 1$$

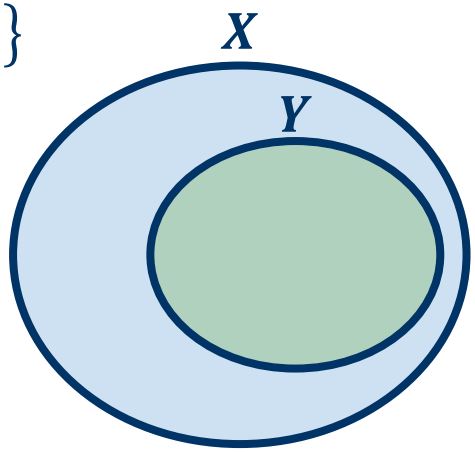
This is k-criterion!

Appendix: Universally undistinguishable mutant detection problem

- **Two mutants m_x and m_y are *universally undistinguishable mutants* when $\forall t \in T, d(t, p_o, m_x) = d(t, p_o, m_y)$.**

$$X = \{(m_x, m_y) | \forall t \in T, d(t, p_o, m_x) = d(t, p_o, m_y)\}$$

$$Y = \{(m_x, m_y) | \forall t \in T, d(t, m_x, m_y) = 0\}$$



Essentially the same as the equivalent mutant detection problem

$$\forall t \in T, d(t, p_o, m) = 0$$

Many practical approximations[1][2][3]

[1] L. Madeyski, W. Orzeszyna, R. Torkar, and M. Jo´zala, "Overcoming the equivalent mutant problem: A systematic literature review and a comparative experiment of second order mutation," *IEEE Transactions on Software Engineering (TSE)*, vol. 40, no. 1, pp. 23–42, 2014.
[2] M. Papadakis, Y. Jia, M. Harman, and Y. Le Traon, "Trivial compiler equivalence: A large scale empirical study of a simple, fast and effective equivalent mutant detection technique," in *Proceedings of the IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, vol. 1. IEEE, 2015, pp. 936–946.
[3] M. Kintis and N. Malevris, "Medic: A static analysis framework for equivalent mutant identification," *Information and Software Technology*, vol. 68, pp. 1–17, 2015.