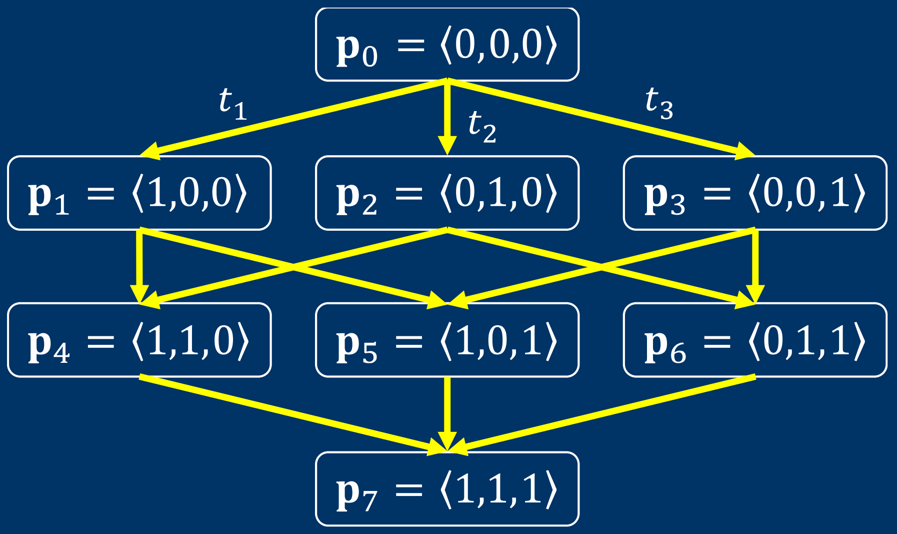
A Theoretical Framework for Understanding Mutation-Based Testing Methods

Donghwan Shin and Doo-Hwan Bae

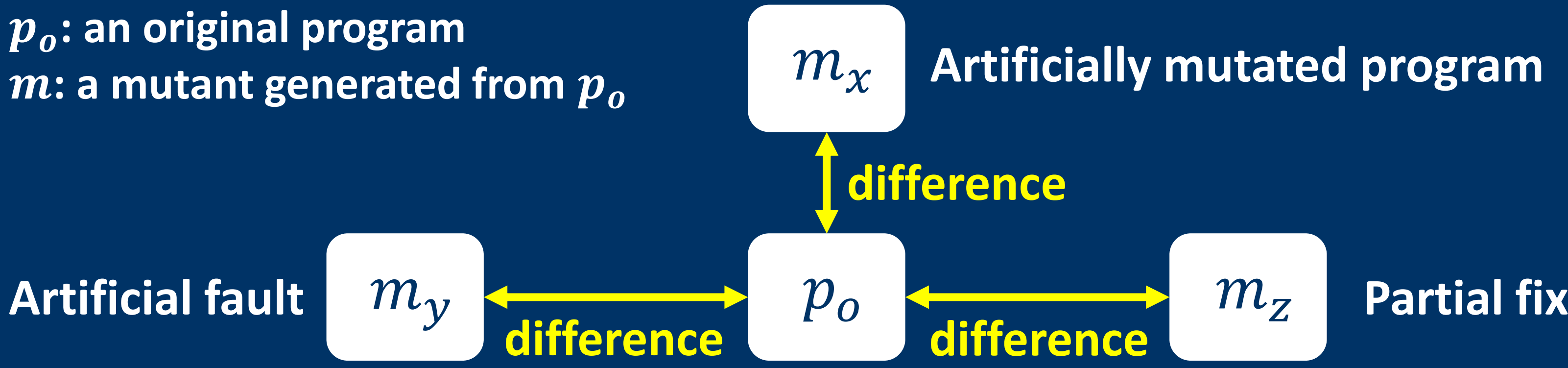
KAIST, South Korea

@ ICST 2016



Mutation-based testing has been widely studied for addressing various testing problems

p_o: an original program
m: a mutant generated from p_o

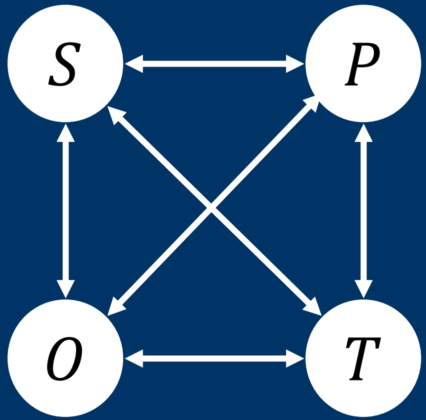


Systematically generate m from p_o and use the differences between them

- Test set selection
- Fault localization
- Program repair

There is a solid theoretical framework for general testing process – “testing system”

Fundamental testing factors and their relationships[1][2]



- P is a set of programs
- S is a set of specifications
- T is a set of tests
- O is a set of oracles

P attempts to implement S

T is designed to consider S and P

O determines the correctness of P for T

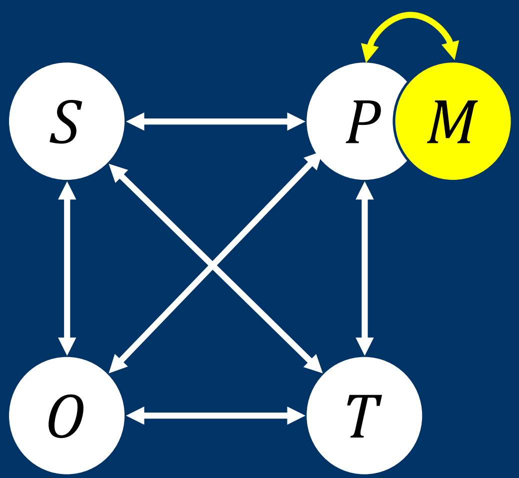
…

Such theoretical framework facilitates a clear understanding of the essence of complex problems

[1] J. S. Gourlay, “A mathematical framework for the investigation of testing,” Software Engineering, IEEE Transactions on, no. 6, pp. 686-709, 1983.
[2] M. Staats, M. W. Whalen, and M. P. E. Heimdahl, “Programs, tests, and oracles: the foundations of testing revisited,” in Proceedings of ICSE 2011, pp. 391-400.

Surprisingly little attention has been paid to the theoretical framework of mutation-based testing

- Even the testing system is not adequate to mutation-based testing because it focuses on the correctness.



- S is a set of specifications
- P is a set of programs
- T is a set of tests
- O is a set of oracles

P attempts to implement S

T is designed to consider S and P

O determines the correctness of P for T

…

How to formally describe the behavioral differences between programs and its mutants in testing?

Key idea: a new testing factor for difference

- **In the testing system, an oracle o implies the correctness of a program p for a test t as follows:**

$$o(t,p) = \begin{cases} 1\ (true), & \text{if } p \text{ is correct for } t \\ 0\ (false), & \text{otherwise} \end{cases}$$

Let us define a new testing factor X
to imply the difference between two programs for a test

$$X(t, p_x, p_y)$$

A theoretical framework for the difference in mutation-based testing

- **In this talk:**
  - Define and extend a new testing factor to formalize the "difference-based testing framework"
- **What you can do using this framework:**
  - Formally describe the behavioral differences of programs given a set of tests.
  - Quantitatively represent and analyze the behavioral differences in a multi-dimensional space.
  - Guide to understand mutation-based testing methods.

Outline

- **Test differentiator**
  - A new testing factor for the notion of difference
- **d-vector**
  - Extended differentiator for a set of tests
- **Position**
  - Redefined d-vector in a multi-dimensional space
- **Position deviance relation**
  - Formal relation between positions
- **Position Deviance Lattice**
  - Graphical model for positions and its deviance relation
- **Applications**

Outline

- **Test differentiator**
  - A new testing factor for the notion of difference
- **d-vector**
  - Extended differentiator for a set of tests
- **Position**
  - Redefined d-vector in a multi-dimensional space
- **Position deviance relation**
  - Formal relation between positions
- **Position Deviance Lattice**
  - Graphical model for positions and its deviance relation
- **Applications**

Test differentiator: a new testing factor (1/2)

Def. 1: A **test differentiator d** is a function such that

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

Example

$d(t, p_o, m) = 1$ "t **detects the difference between $\boldsymbol{p_o}$ and $\boldsymbol{m}$**"

"The test *kills* the mutant"

Test differentiator: a new testing factor (2/2)

- It clarifies tricky concepts in mutation-based testing.

> A test t detects a fault in p_o
>
> A test t kills a mutant m

p_s: the projection of the specification (i.e., true requirements of p_o)

- It shows how d is important in mutation-based testing.

Weak mutation ⟷ Strong mutation

d observes internal values

d observes only output

Outline

- Test differentiator
  - A new testing factor for the notion of difference
- d-vector
  - Extended differentiator for a set of tests
- Position
  - Redefined d-vector in a multi-dimensional space
- Position deviance relation
  - Formal relation between positions
- Position Deviance Lattice
  - Graphical model for positions and its deviance relation
- Applications

d-vector: extended d as a vector form

Def. 2: A d**-vector** $\boldsymbol{d}$ is an n-dimensional vector, such that

$$\mathbf{d}(\mathbf{t}, p_x, p_y) = \langle d(t_1, p_x, p_y), \dots, d(t_n, p_x, p_y) \rangle$$

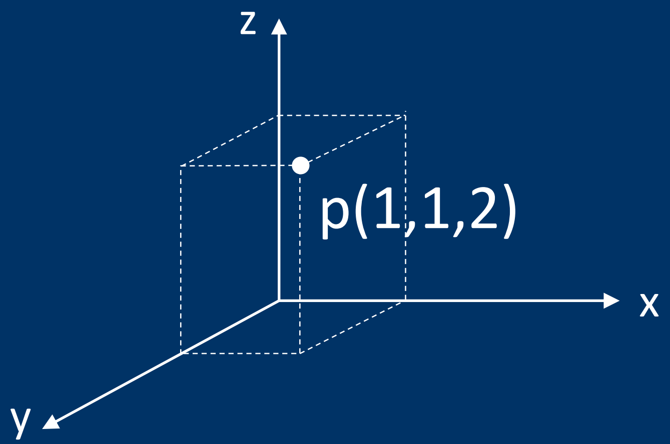
for a collection of tests $\mathrm{t} = \langle t_1, t_2, \cdots, t_n \rangle \in \boldsymbol{T}^n$.

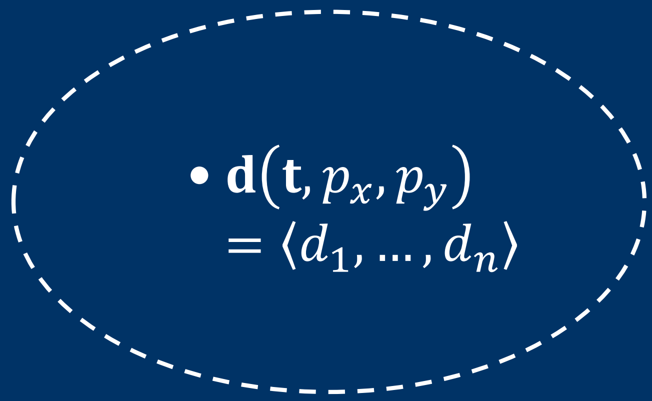
Example

Test	Behavior			d-vector = behavioral difference		
t_4	D	A	D			

d-vector concisely describes the behavioral differences of programs for a set of tests

d-vector = position in a multi-dimensional space





- **A vector represents a point in a multi-dimensional space.**
- **The position of a point is relative to the origin.**

- **We can think of a d-vector as the representation of a position in a space.**

Position and its space from a d-vector

Def. 3: The **position of a program p_x** relative to another program p_r in a multi-dimensional space corresponding to a set of tests t is

$$\mathbf{d}(\mathrm{t}, p_r, p_x) = \mathbf{d}^{\mathbf{t}}_{p_r}(p_x).$$

- The space is induced by (t, p_r, d).
  - ✓ $\mathbf{t} = \langle \boldsymbol{t_1}, \cdots, \boldsymbol{t_n} \rangle$ corresponds to **the $\boldsymbol{n}$-dimensions**.
  - ✓ p_r corresponds to **the origin** of the space.
  - ✓ d corresponds to **the notion of difference** in the space.
- A position of $\boldsymbol{p_x}$ relative to the origin $\boldsymbol{p_r}$ implies the behavioral difference between $\boldsymbol{p_x}$ and $\boldsymbol{p_r}$ for a set of tests t.

Outline

- Test differentiator
  - A new testing factor for the notion of difference
- d-vector
  - Extended differentiator for a set of tests
- Position
  - Redefined d-vector in a multi-dimensional space
- **Position deviance relation**
  - **Formal relation between positions**
- Position Deviance Lattice
  - Graphical model for positions and its deviance relation
- Applications

Position deviance relation (1/3)

- Add more tests → increasing dimensions
  → make positions deviant from its origin

Example

$\mathbf{p_y}$ ***is deviant from*** $\mathbf{p_x}$ by t_{i+1}

$\mathbf{t} = \{\cdots, t_i, \cdots\}$

$\mathbf{p_x} = \langle \cdots, 0, \cdots \rangle$

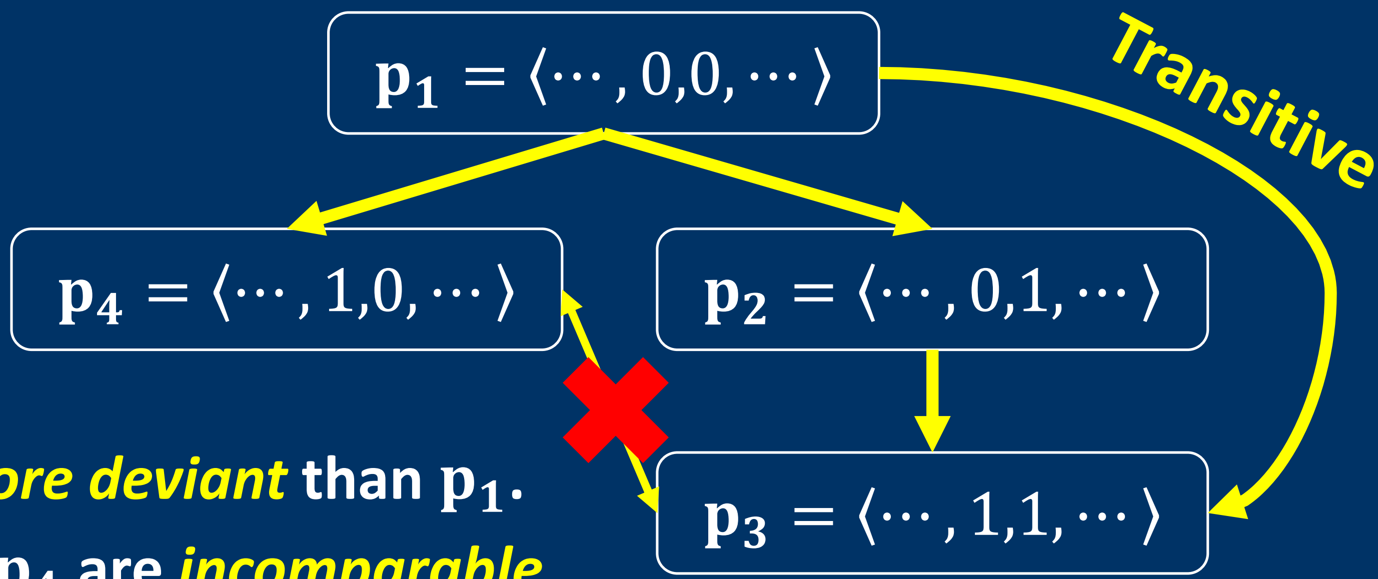
$\mathbf{t} = \{\cdots, t_i, t_{i+1}, \cdots\}$

$\mathbf{p_x} = \langle \cdots, 0, 0, \cdots \rangle$

$\mathbf{p_y} = \langle \cdots, 0, 1, \cdots \rangle$

Position deviance relation (2/3)

- Forms a *partial order* based on transitivity.



✓ $\mathbf{p_3}$ is ***more deviant*** than $\mathbf{p_1}$.

✓ $\mathbf{p_3}$ and $\mathbf{p_4}$ are ***incomparable***.

Position deviance relation (3/3)

- If $p_r = p_o$, the deviance relation on positions *implies* the dynamic subsumption* on mutants at the positions.

$$\mathbf{p_o} = \langle 0, \cdots, 0 \rangle = \{p_o\}$$

$$\downarrow$$

$$\mathbf{p_x} = \langle \cdots, 0,0, \cdots \rangle = \{m_x\}$$

$$\downarrow$$

Dynamic subsumption

If m_x is killed, then m_y must be killed.

$$\mathbf{p_y} = \langle \cdots, 0,1, \cdots \rangle = \{m_y\}$$

* Ammann, Paul, Marcio E. Delamaro, and Jeff Offutt. "Establishing theoretical minimal sets of mutants." ICST 2014.

Outline

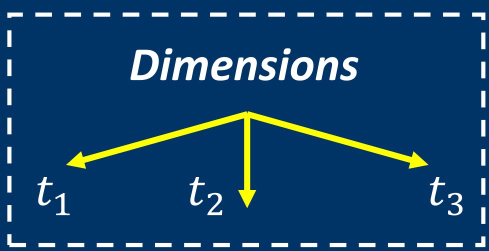
- Test differentiator
  - A new testing factor for the notion of difference
- d-vector
  - Extended differentiator for a set of tests
- Position
  - Redefined d-vector in a multi-dimensional space
- Position deviance relation
  - Formal relation between positions
- **Position Deviance Lattice**
  - **Graphical model for positions and its deviance relation**
- Applications

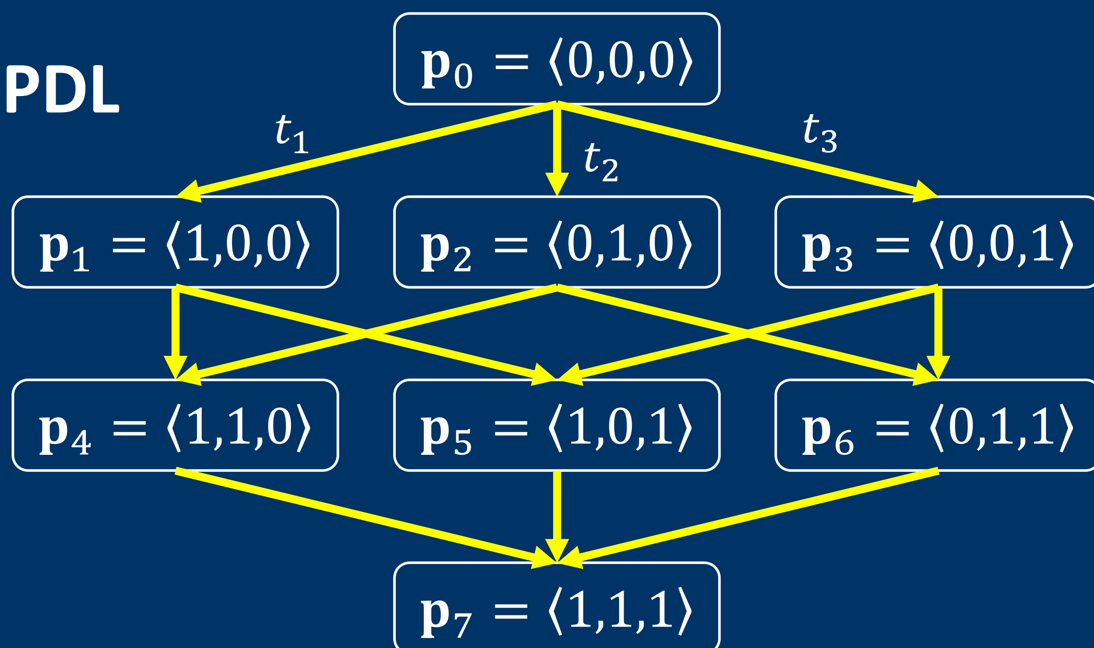
Position Deviance Lattice (PDL) (1/4)

- **It shows the positions with their deviance relation.**
  - **Each dimension (=test) has only two positions, 0 or 1.**
  - **The number of positions in a n-dimensional space = 2^n.**

Example

3-dimensional PDL

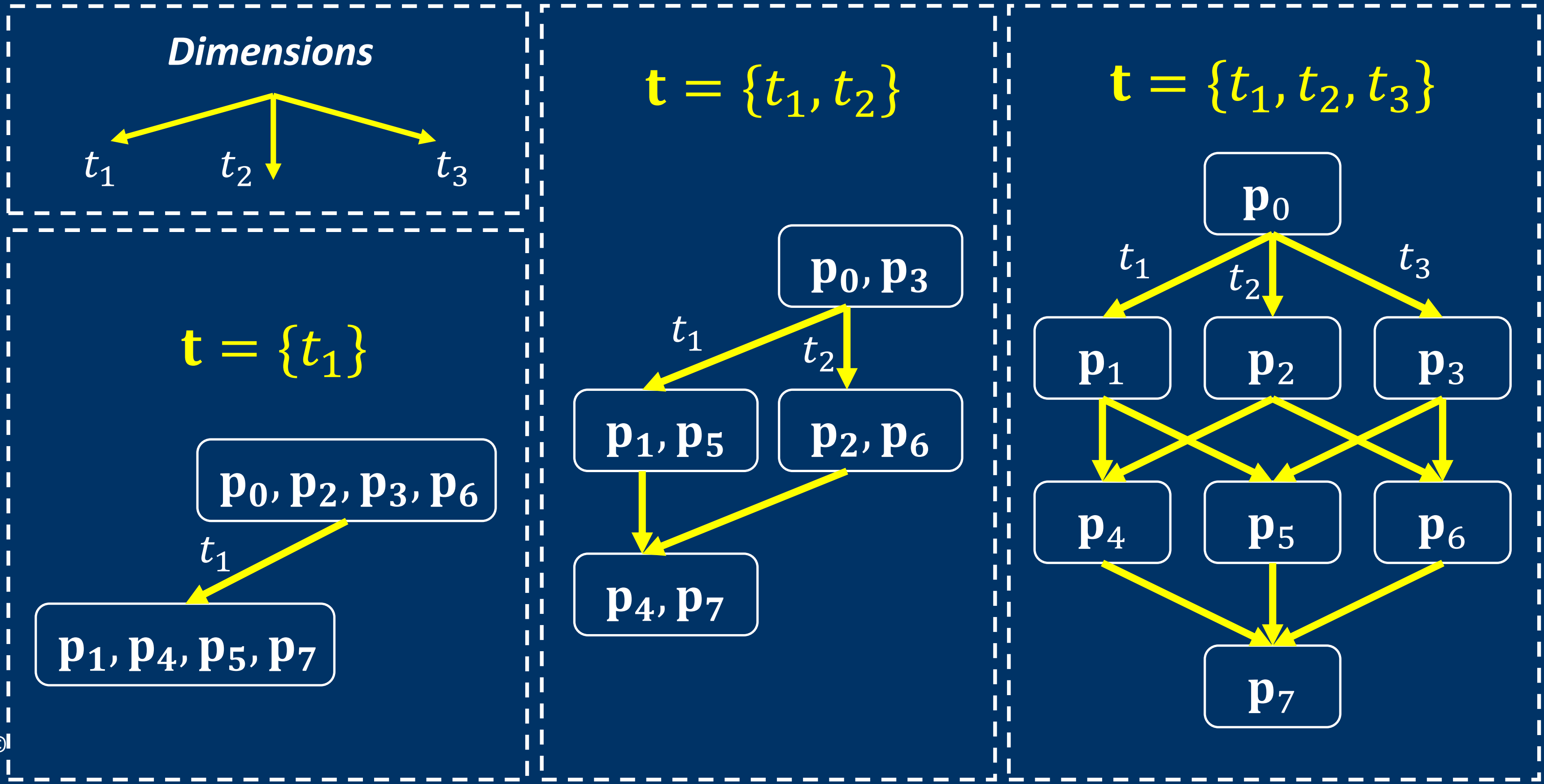




Position Deviance Lattice (PDL) (2/4)

- It growths as more tests added.

Example

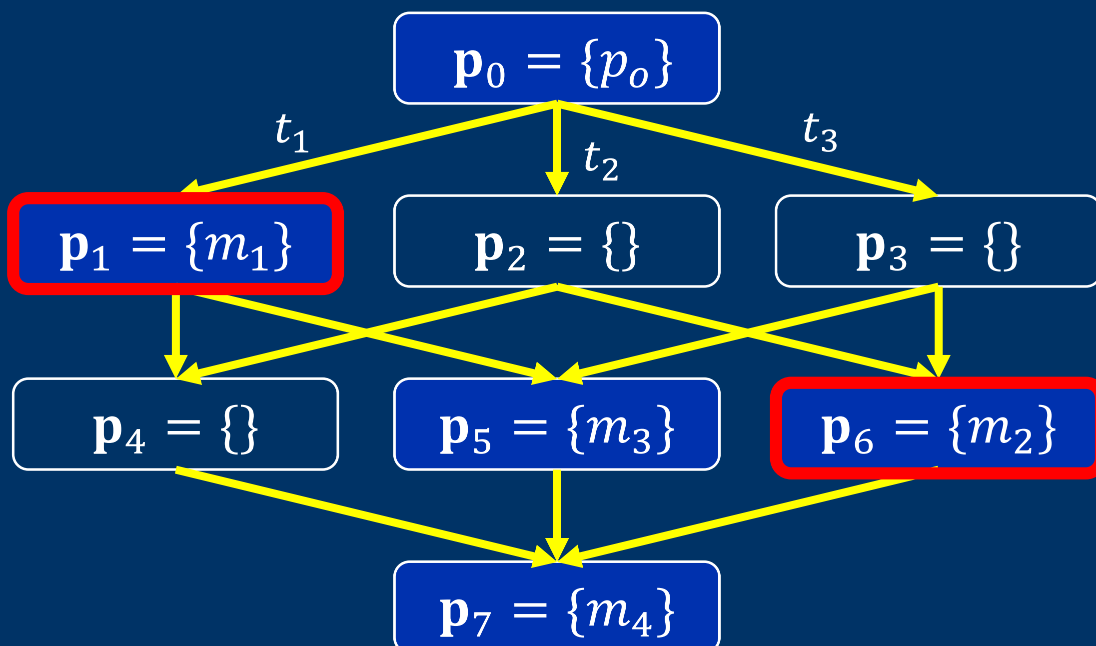


Outline

- Test differentiator
  - A new testing factor for the notion of difference
- d-vector
  - Extended differentiator for a set of tests
- Position
  - Redefined d-vector in a multi-dimensional space
- Position deviance relation
  - Formal relation between positions
- Position Deviance Lattice
  - Graphical model for positions and its deviance relation
- Applications

Minimal set of mutants in PDL (1/2)

- In PDL where $p_r = p_o$, mutants in "the least deviant" positions form the minimal set of mutants.



Minimal set of mutants in PDL (2/2)

- **What is the maximum bound of the minimal set of mutants for n tests?**
  - **Key: if two mutants are at two comparable positions, then one must be dynamically subsumed by another.**
  - ***Sperner's theorem* says that the maximum number of incomparable nodes in an n-dim lattice is given as follows:**

$$\max(|M_{minimal}|) = \binom{n}{\lfloor n/2 \rfloor}$$

(E.g.) If we have 10 tests, # of minimal mutants $\leq \binom{\mathbf{10}}{\mathbf{5}} =$ 252

Understanding and extending the mutation adequacy criterion using PDL

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

$TS = \{\}$

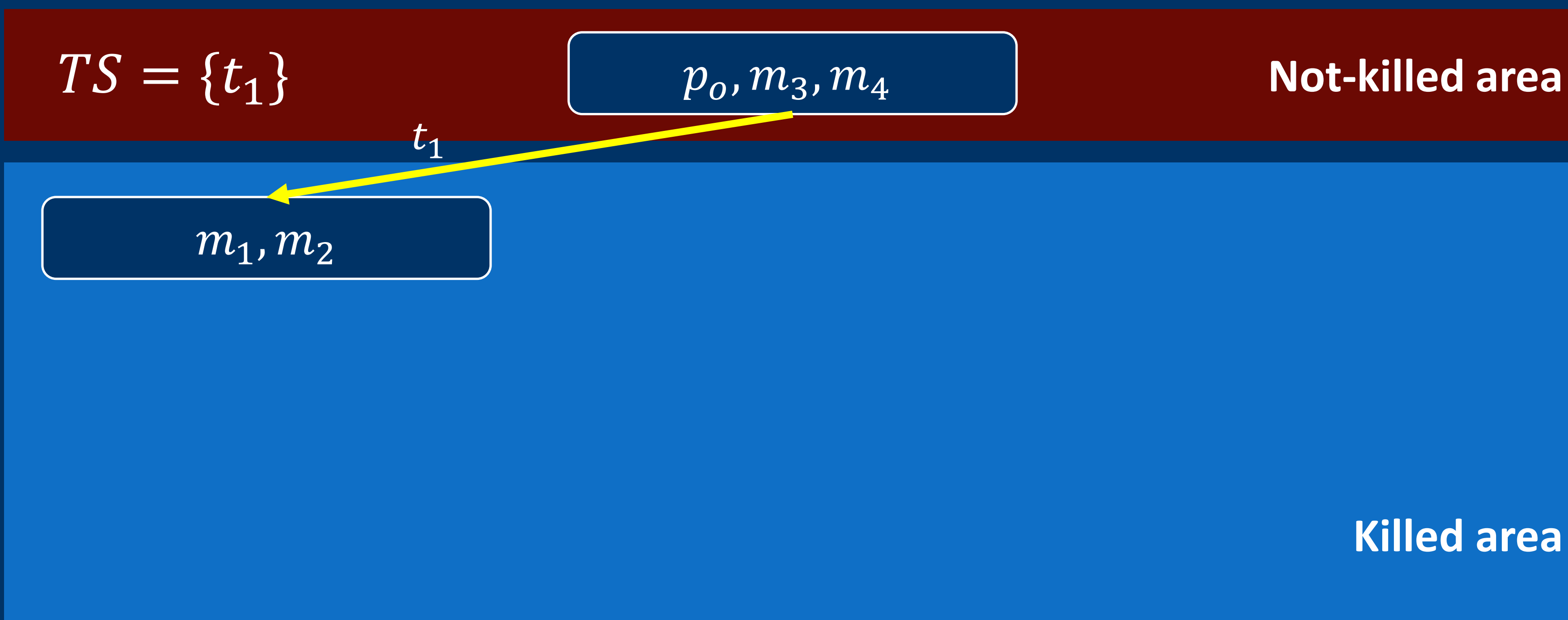
p_o, m_1, m_2, m_3, m_4

Not-killed area

Killed area

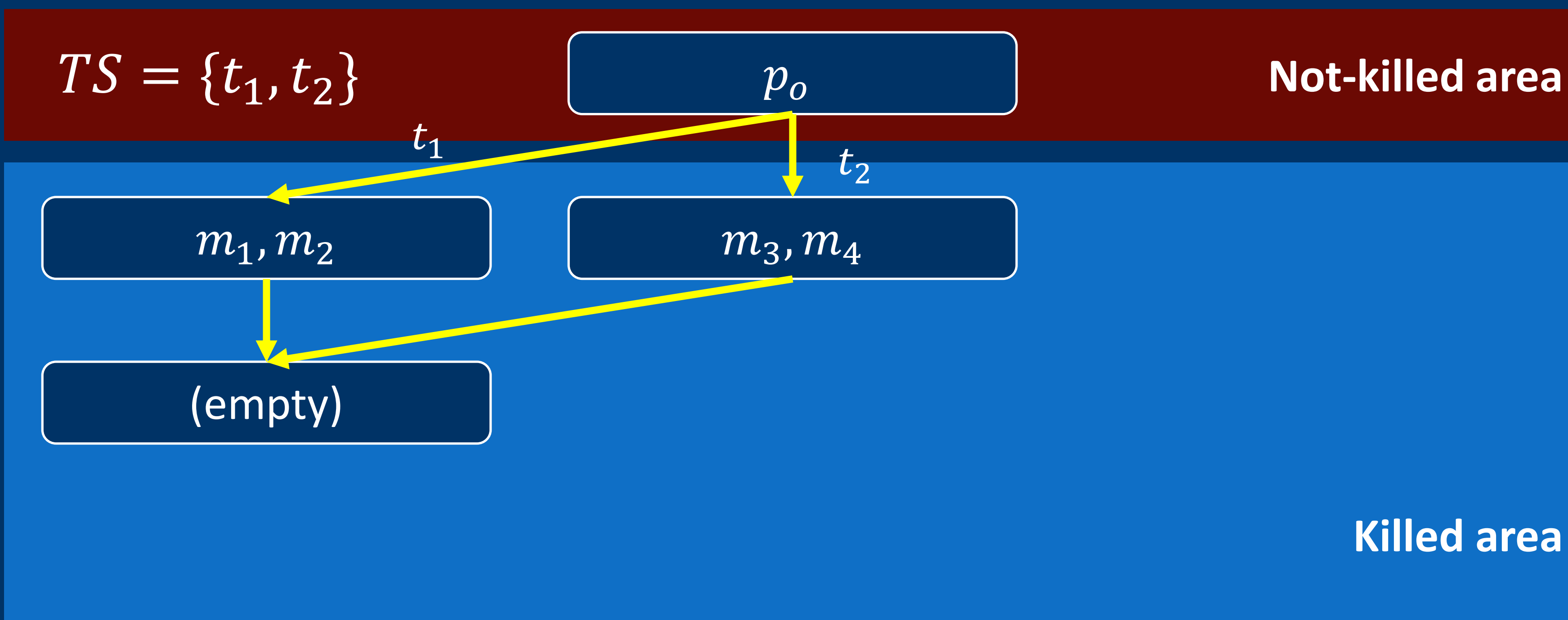
Understanding and extending the mutation adequacy criterion using PDL

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



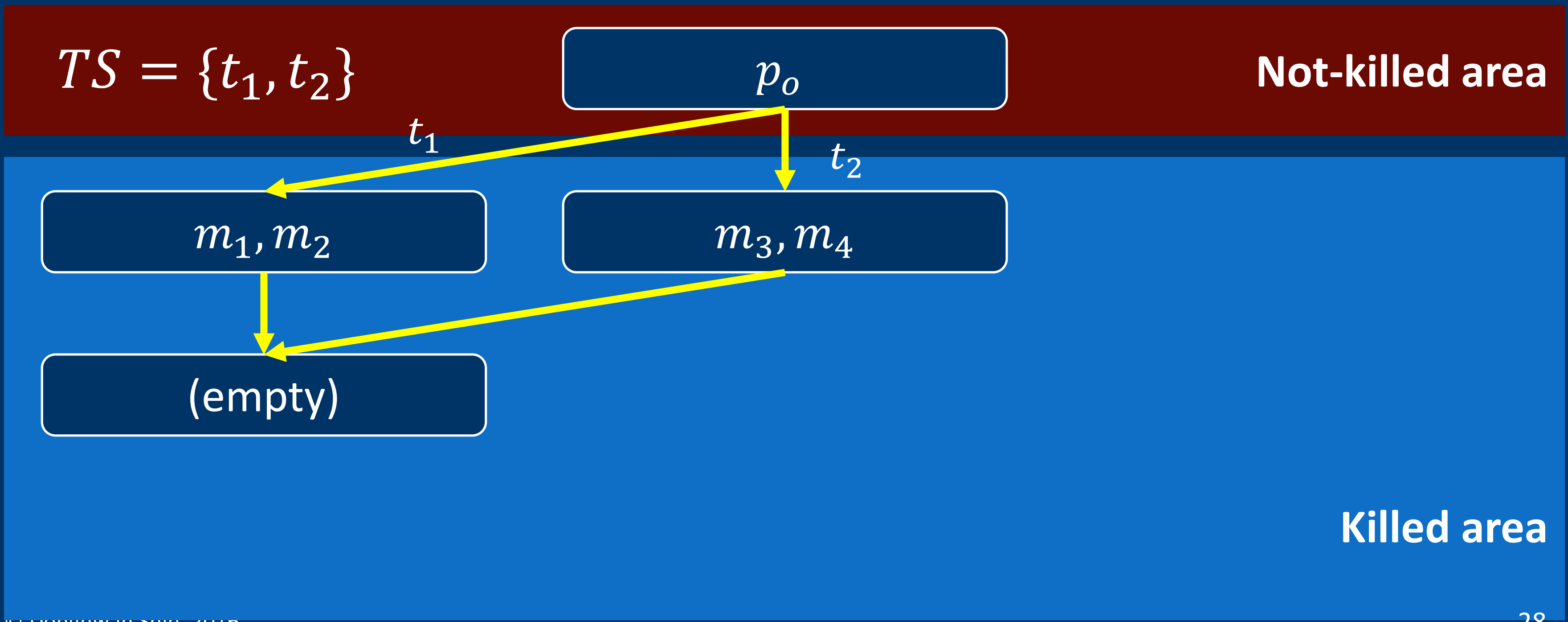
Understanding and extending the mutation adequacy criterion using PDL

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



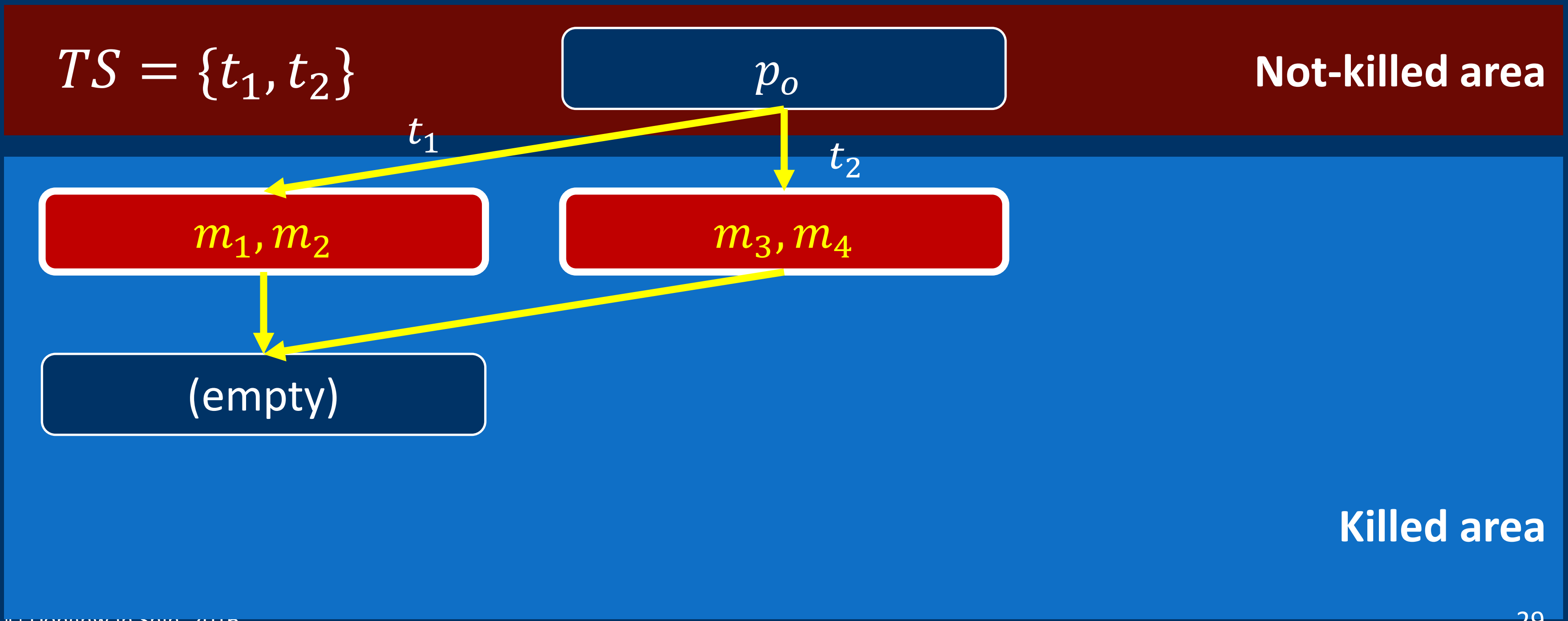
Understanding and extending the mutation adequacy criterion using PDL

- **Traditional mutation adequacy criterion**
  - **A test suite that distinguishes the positions of mutants from the position of p_o will likely detect real faults.**



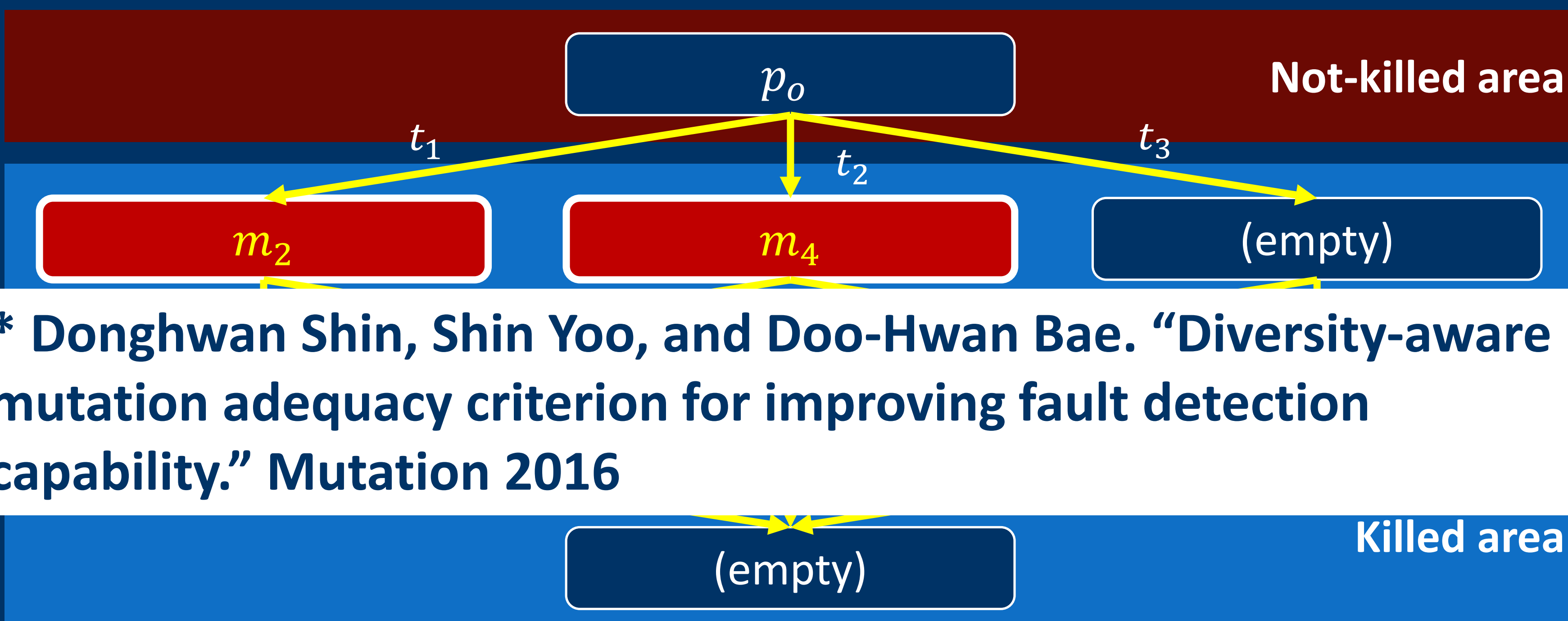
Understanding and extending the mutation adequacy criterion using PDL

- **Claim for the existing mutation adequacy criterion**
  - In the "killed area", there are still several mutants which are not distinguished in terms of their positions.



Understanding and extending the mutation adequacy criterion using PDL

- **Distinguishing more mutants increases fault detection***
  - A test suite that distinguishes the positions of mutants **from each other** will likely detect real faults.



* Donghwan Shin, Shin Yoo, and Doo-Hwan Bae. "Diversity-aware mutation adequacy criterion for improving fault detection capability." Mutation 2016

Conclusion

- **Correctness-based think → difference-based think**
  - **Incorrect programs seem useless, while different programs seems meaningful.**
- **PDL may guide you to consider difference-based think.**