FONDAZIONE BRUNO KESSLER

SE SOFTWARE ENGINEERING

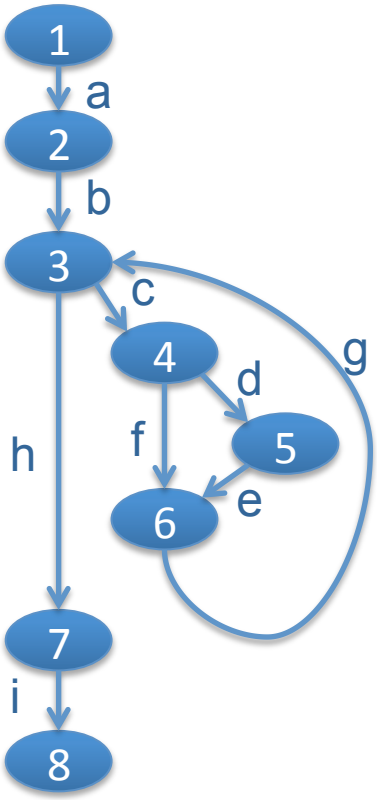
DSE+SBST: Marriage or Divorce

Paolo Tonella

Fondazione Bruno Kessler

Trento, Italy

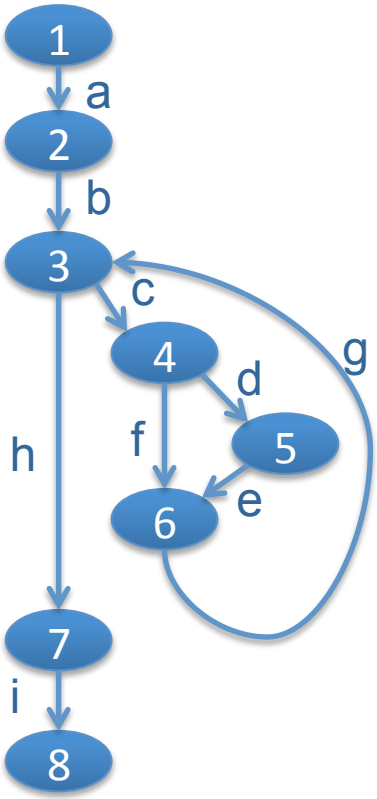
Different goals



SBST = {a, b, c, d, e, f, g, h, i}

DSE= {<a, b, h, i>,
<a, b, c, f, g, h, i>,
<a, b, c, d, e, g, h, i>,
<a, b, c, d, e, g, c, f, g, h, i>,
<a, b, c, d, e, g, c, d, e, g, h, i>,
…}

Different goals



SBST = {a, b, c, d, e, f, g, h, i}

DSE= {<a, b, h, i>,
<a, b, c, f, g, h, i>,
<a, b, c, f, g, c, f, g, h, i>,
<a, b, c, f, g, c, f, g, c, f, g, h, i>,
…}

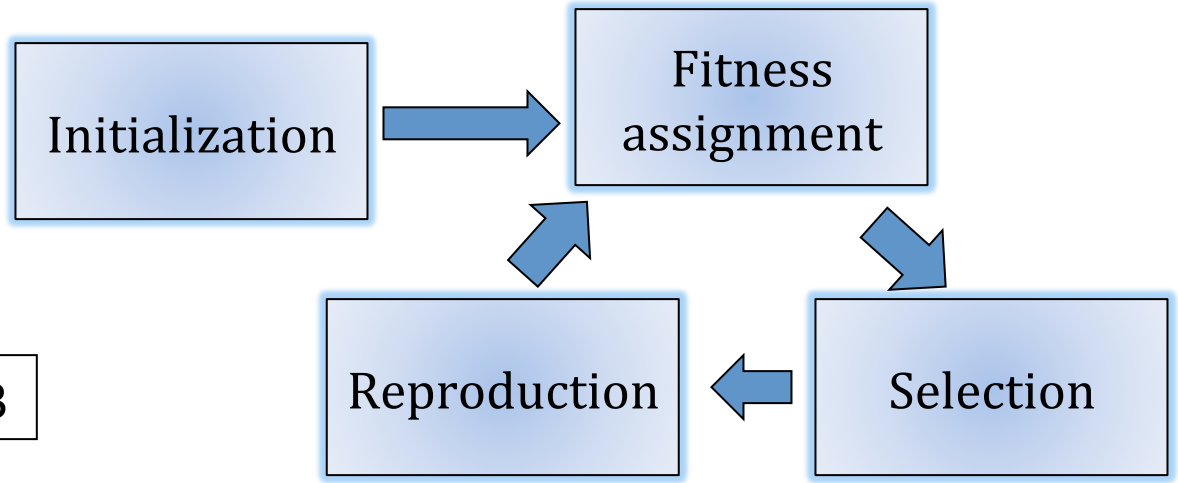
Different weaknesses and strengths

	Weaknesses	Strengths
DSE	• Loops • Black box functions • Non linear constraints • Complex data structures • Divergences • Reflection	• Exploration strongly guided by path condition • Few executions (fitness evaluations) required
SBST	• Flat (non-guiding) fitness functions • Deceptive fitness functions • Many fitness evaluations (executions) required	Robust w.r.t complex/unknown program semantics (e.g., black box functions, non linear expressions, complex data structures, reflection)

Existing combinations

- DSE as additional genetic operator [**M&F'11,GF&A'13**]
- Alternation between DSE and SBST [**I&X'08**]
- Fitness used to select which path to explore in DSE [**XTH&S'09**]
- Symbolic execution based fitness in SBST [**BHHLMT&V'11**]

Additional genetic operator



Ch1:

1	5	-1	0	3

pc1: C1 && C2 && **C3** && C4

pc1': C1 && C2 && !**C3**

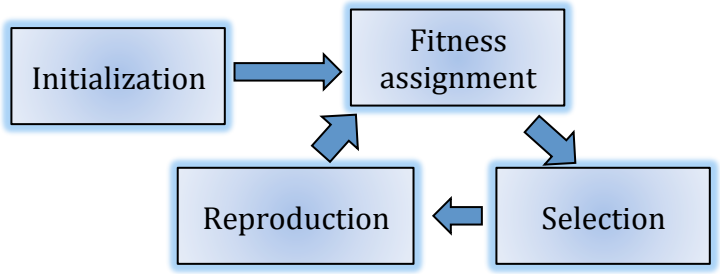
Ch1':

1	0	2	1	1

- Mutation
- Crossover
- **DSE based mutation**

[**M&F'11**] Jan Malburg, Gordon Fraser. *Combining search-based and constraint-based testing*. Proc. of Automated Software Engineering (ASE), pp. 436-439, 2011.

Additional genetic operator



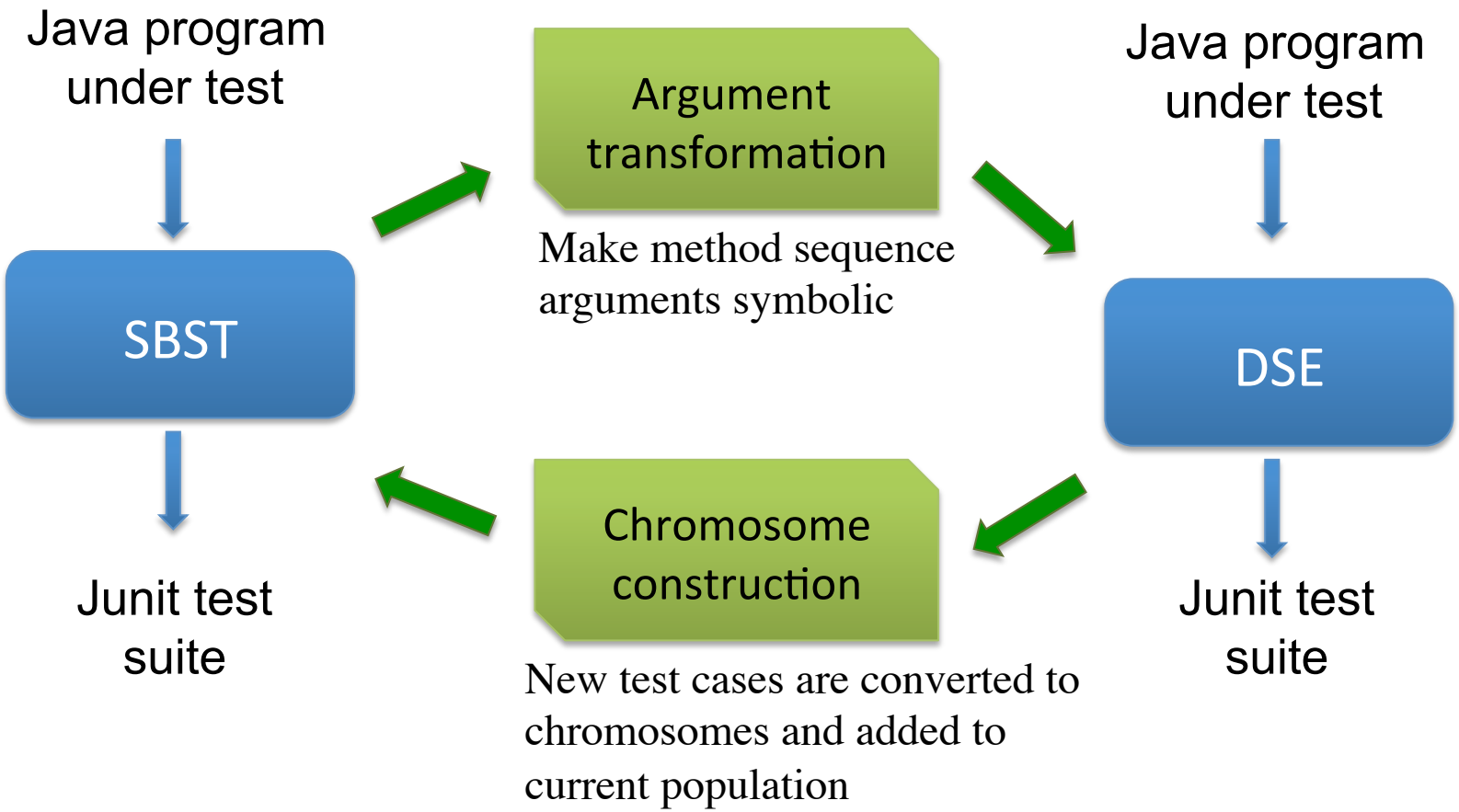
- Mutation [f'>f]-> DSE
- Crossover

1. **On which individuals is DSE applied?**
2. **When is it applied?**
3. **How is it applied?**

1. **Individuals for which primitive mutation affects the fitness.**
2. **DSE is applied with probability P (suggested value = 100%).**
3. **Individuals produced by DSE are kept only if they improve the fitness**

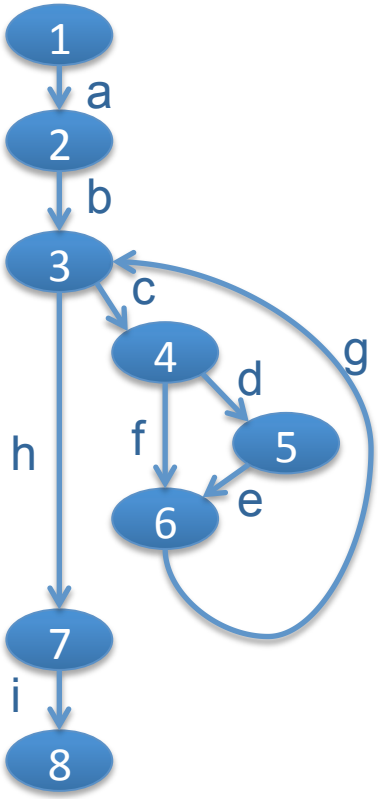
[**GF&A'13**] Juan Pablo Galeotti, Gordon Fraser, Andrea Arcuri. *Improving Search-based Test Suite Generation with Dynamic Symbolic Execution*. Proc. of the 24th International Symposium on Software Reliability Engineering (ISSRE), 2013.

Alternation



[**I&X'08**] Kobi Inkumsah, Tao Xie. *Improving Structural Testing of Object-Oriented Programs via Integrating Evolutionary Testing and Symbolic Execution*. Proc. of Automated Software Engineering (ASE), pp. 297-306, 2008.

Path selection

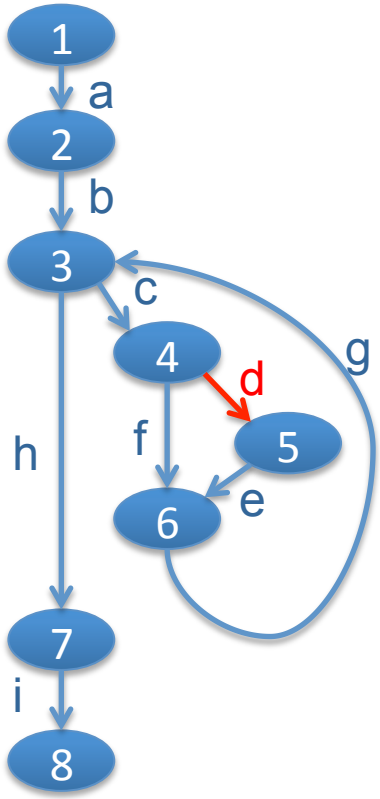


DSE= {<a, b, h, i>, // TC1
<a, b, c, f, g, h, i>, // TC2
<a, b, c, f, g, c, f, g, h, i>} // TC3

- Which test case shall be selected for branch flipping?
- Which branch shall be flipped?

[**XTH&S'09**] Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. *Fitness-guided path exploration in dynamic symbolic execution*. Proc. of the International Conference on Dependable Systems and Networks (DSN), pp. 359-368, 2009.

Path selection



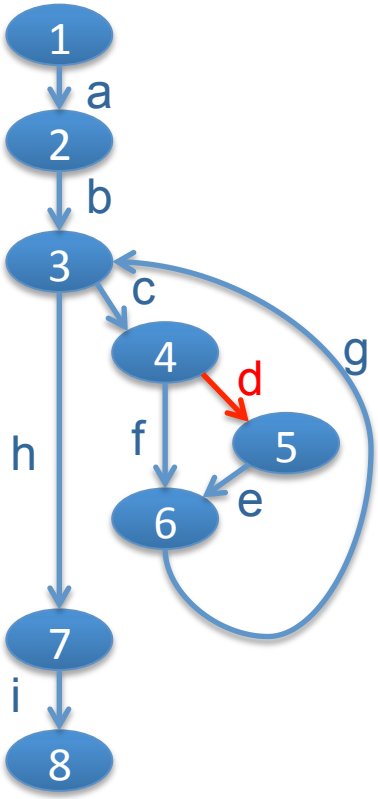
DSE= {<a, b, h, i>, // TC1
<a, b, c, f, g, h, i>, // TC2
<a, b, c, f, g, c, f, g, h, i>} // TC3

- Which test case shall be selected for branch flipping?

BranchDistance(TC2, d) = 0.8
BranchDistance(TC3, d) = min(0.8, 0.5) = 0.5

[**XTH&S'09**] Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. *Fitness-guided path exploration in dynamic symbolic execution*. Proc. of the International Conference on Dependable Systems and Networks (DSN), pp. 359-368, 2009.

Path selection



DSE= {<a, b, h, i>, // TC1
<a, b, c, f, g, h, i>, // TC2
<a, b, c, f, g, c, f, g, h, i>} // TC3

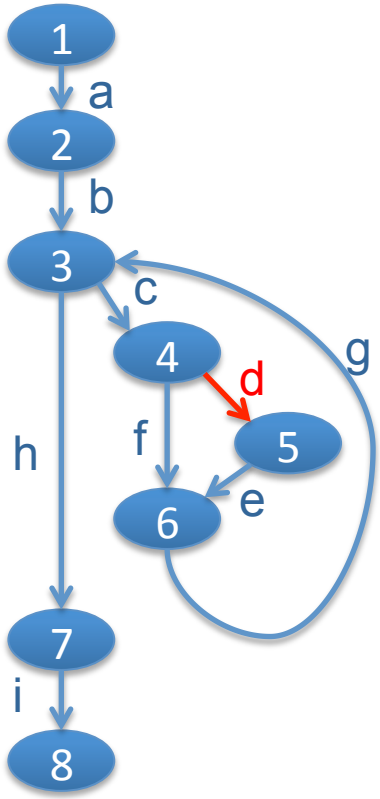
$3T_1$ $4F_1$ $3T_2$ $4F_2$ $3F_1$

- Which branch shall be flipped?

Solve(PC3[C$4F_1$ ➔ !C$4F_1$]) = *UNSAT*
Solve(PC3[C$4F_2$ ➔ !C$4F_2$]) = *UNSAT*
FitnessGain(3F) = (1+1+0.3)/3 = 0.76
FitnessGain(3T) = -0.76

[**XTH&S'09**] Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. *Fitness-guided path exploration in dynamic symbolic execution*. Proc. of the International Conference on Dependable Systems and Networks (DSN), pp. 359-368, 2009.

New fitness function

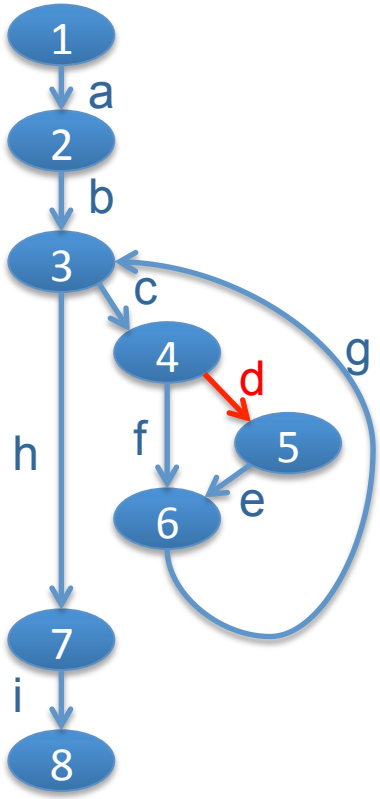


PathExpression(f, d) = fg(cfg)*cd

P1 = fgcd
P2 = fgcfgcd
P3 = fgcfgcfgcd
P4 = fgcfgcfg **D[cfg$^+$]** cd

[**BHHLMT&V'11**] Arthur I. Baars, Mark Harman, Youssef Hassoun, Kiran Lakhotia, Phil McMinn, Paolo Tonella, Tanja E. J. Vos. *Symbolic search-based testing*. Proc. of ASE, pp. 53-62, 2011.

New fitness function



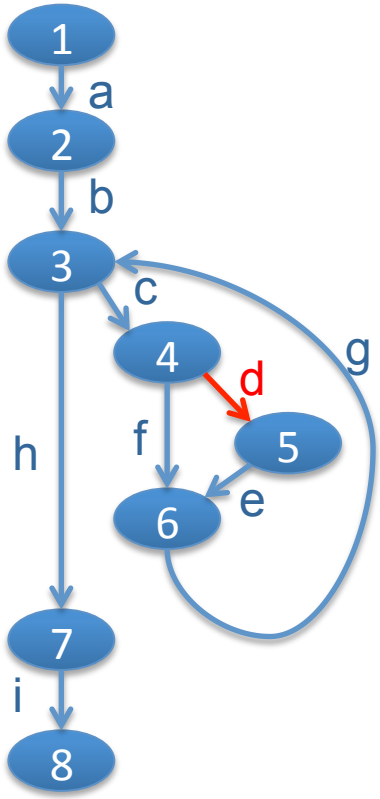
P1 = fgcd
PC1 = C3 ∧ !C4
FF1 = BD(C3) + BD(!C4)

P2 = fgcfgcd
PC2 = C3 ∧ C3' ∧ !C4
FF2 = BD(C3) + BD(C3') + BD(!C4)

P3 = fgcfgcfgcd
PC3 = C3 ∧ C3' ∧ C3'' ∧ !C4
FF3 = BD(C3) + BD(C3') + BD(C3'') + BD(!C4)

[**BHHLMT&V'11**] Arthur I. Baars, Mark Harman, Youssef Hassoun, Kiran Lakhotia, Phil McMinn, Paolo Tonella, Tanja E. J. Vos. *Symbolic search-based testing*. Proc. of ASE, pp. 53-62, 2011.

New fitness function



P4 = fgcfgcfg **D[cfg$^+$]** cd

PC4 = C3 ∧ C3' ∧ C3'' ∧ D[C3] ∧ !C4

FF4 = BD(C3) + BD(C3') + BD(C3'') + 1 + BD(!C4)

Fitness(TC, f, d) = *min*(FF1[TC], FF2[TC], FF3[TC], FF4[TC])

StdFitness(TC, f, d) = BD(!C4)

[**BHHLMT&V'11**] Arthur I. Baars, Mark Harman, Youssef Hassoun, Kiran Lakhotia, Phil McMinn, Paolo Tonella, Tanja E. J. Vos. *Symbolic search-based testing*. Proc. of ASE, pp. 53-62, 2011.

Comparison of existing combinations

Paper	Switching vs. Enhancing	Overcome limitation	Algorithmic change
[**M&F'11, GF&A'13**]	S	Stagnation/slow convergence	New genetic operator
[**I&X'08**]	S	Long time required for parameter value generation	Meta-level algorithm for switching
[**XTH&S'09**]	E (DSE)	Path exploration preventing (branch) coverage	Test case and branch selection
[**BHHLMT&V'11**]	E (SBST)	Standard fitness, accounting only for shortest path to target	New fitness function

- DSE as additional genetic operator [**M&F'11, GF&A'13**]
- Alternation between DSE and SBST [**I&X'08**]
- Fitness used to select which path to explore in DSE [**XTH&S'09**]
- Symbolic execution based fitness in SBST [**BHHLMT&V'11**]

Beyond existing combinations

- **Better alternation**
  - [loop, black-box, non-linear, complex data, reflection] ➔ SBST
  - [fitness stagnation, excessive executions] ➔ DSE
- **Deeper integration**
  - Reconcile different goals
  - Unify constraint solving and fitness function evaluation