An Algorithm for Generating t-wise Covering Arrays from Large Feature Models

Martin Fagereng Johansen
Øystein Haugen
Franck Fleurey

SPLC 2012 – Salvador, Brazil
## Example Product Line: The Eclipse IDEs

<table>
<thead>
<tr>
<th></th>
<th>Java EE</th>
<th>Java EE</th>
<th>C/C++ Linux</th>
<th>RCP/RAP</th>
<th>Modeling</th>
<th>Reporting</th>
<th>Parallel</th>
<th>Scout</th>
<th>Testers</th>
<th>Javascript</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCP/Platform</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CVS</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>EGit</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>EMF</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GEF</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>JDT</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Mylyn</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Web Tools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Linux Tools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Java EE Tools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>XML Tools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>RSE</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>EclipseLink</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>PDE</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>DataTools</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>CDT</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>BIRT</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>GMF</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>PTP</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>MDT</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Scout</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Jubula</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>RAP</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>WindowBuilder</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Maven</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Constraints Between Features

356,352 possible products
Product Line Verification

- How do we gain confidence that any valid product works?
Faulty Features

- Unit tests may find faults inside a single feature.
  - n test suites required for a product line with n features.
- What about faulty cooperation between features?
  - What if they interact incorrectly?
Interaction Faults

- 2-wise interaction fault
  - reproducible by including 2 specific features
  - the others do not matter
Interaction Faults

- 3-wise interaction fault
  - reproducible by including 3 specific features
  - the others do not matter
Empirics Show:

- Kuhn et al. 2004:
  - Most bugs can be attributed to the interaction of a few features.
Covering Arrays

- Mathematical property:
  - Only a few products needed to cover all simple interactions

- Other examples (pair-wise testing):
  - For the "e-shop product line" with 287 features: 21 products
  - For the Linux kernel with almost 7,000 features: 480 products
Combinatorial Interaction Testing (CIT)

1. Generate a covering array
   - Can be reused until the feature model is changed
2. Build each product
3. Apply a single system testing technique to each product

Note: CIT was originally intended for single system testing
   - Covering arrays over input instead of interactions.

<table>
<thead>
<tr>
<th>Feature/Product</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>EclipseIDE</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Team</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVS</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVS_Over_SSH</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVS_Over_SSH2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVN</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subversive</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subclipse</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subclipse_1.x</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subclipse_1.6.x</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GIT</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EclipseFileSystem</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Local</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Background

Our MODELS 2011 paper concludes:

- Covering array generation is tractable in practice.
  - Difficult to satisfy FMs imply no products to sell, which is absurd.
- An efficient algorithm was not provided.
  - 2-wise testing limit: about 500 features
  - 3-wise testing limit: about 200 features

An efficient algorithm is contributed in this paper.

- 2-wise testing
  - Now works for the Linux Kernel feature model (6888 features)
- 3-wise testing
  - Now works for the eCos feature model (1244 features)
  - (An optimized C/C++ implementation + some good hardware should work even for the Linux Kernel feature model.)
Overview of the Algorithm

Implementation Supports
- Simple XML Feature Models (SXFM)
- GUI DSL
- DIMACCS
- CVL (Proposed OMG standard)

CSV-file
Groundwork

Data Structures
- \( a = (\text{feature, included}) \) – an assignment
- \( e = \{a_1, a_2, ..., a_t\} \) – a t-set – a set of t assignments
- \( T_t \) – the set of all t-sets
- \( I_t \) – the set of all invalid t-sets
- \( U_t \) – the set of all valid t-sets (the "universe")
- \( C \) – configuration – a set of assignments, one for each feature
- \( CA_t = \{C_1, C_2, ..., C_x\} \) – a Covering Array of strength t

Equations
- \( |T_t| = 2^t \binom{f}{t} \), i.e. 95 million pair-wise interactions for the Linux kernel
- Empirically, \( |I_t| \ll |U_t| \)
- \( CA_{t-1} \subseteq CA_t \), thus, generating \( CA_{t-1} \) before \( CA_t \) is an option.
Recursive generation of covering arrays of lesser strength

Greedy Loop

- Fit as many as possible
- Remove those covered
- ... and repeat until all interactions are covered

Not parallel

Algorithm 2 \(ICPL(FM, t) : (C_t, I_t)\)

1: if \(t = 1\) then
2: \((C_t, I_t) \leftarrow genCompleteI_1(FM)\)
3: \(T_t \leftarrow FM.genT(1) \setminus I_t\)
4: invalidRemoved \leftarrow true
5: else
6: \((C_t, I_{t-1}) \leftarrow ICPL(FM, t - 1)\)
7: \((T_t, I_t) \leftarrow generateTSets(FM.getT(t), I_{t-1}, C_t)\)
8: invalidRemoved \leftarrow false
9: end if
10: while \(T_t \neq \emptyset\) do
11: \(C \leftarrow genConfiguration(FM, T_t)\)
12: \(C_t \leftarrow C_t \cup \{C\}\)
13: \(CO \leftarrow getCovered(FM, C, T_t)\)
14: \(T_t \leftarrow T_t \setminus CO\)
15: if \((-invalidRemoved) \land (\lceil \log_{10}|CO|\rceil \leq \lceil \log_{10}|FM|\rceil)\) then
16: \((T_t, TempI) \leftarrow genInvalid(FM, T_t)\)
17: \(I_t \leftarrow I_t \cup TempI\)
18: invalidRemoved \leftarrow true
19: end if
20: end while
21: return \((C_t, I_t)\)
Pick an interaction
- Skip if covered

Does it fit the product?
- yes
- no

Make sure all features are assigned

Not parallel
Algorithm 3
- Is the assignment valid?

Algorithm 7
- For all uncovered interactions
- Is the interaction covered?

Algorithm 8
- Pick an interaction
- Check if it is invalid

These are data-parallel sub-algorithms
Compared to Other Tools

- ICPL – our new algorithm
- CASA – Simulated annealing algorithm by Garvin et al.
- MoSo-Polite – algorithm by Oster et al.
- IPOG – algorithm by Lei et al.

Experiment Machine
- Could execute 6 threads in parallel
- 32 GiB RAM
Time Taken to Generate

- Statistic estimates
  - The $c$ in $O(f^c)$ where $f$ is the number of features
Size of Covering Arrays

1-wise

2-wise

3-wise

- ICPL
- CASA
- IPOG*
- MoSo-Polite
# Large Feature Models

<table>
<thead>
<tr>
<th>Feature Model/keys</th>
<th>Features</th>
<th>Constraints (CNF clauses)</th>
<th>SAT time (ms)</th>
<th>1-wise size</th>
<th>1-wise time (s)</th>
<th>2-wise size</th>
<th>2-wise time (s)</th>
<th>3-wise size</th>
<th>3-wise time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.28.6-icse11.dimacs</td>
<td>6,888</td>
<td>187,193</td>
<td>125</td>
<td>25</td>
<td>89</td>
<td>480</td>
<td>33,702</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>freebsd-icse11.dimacs</td>
<td>1,396</td>
<td>17,352</td>
<td>18</td>
<td>9</td>
<td>10</td>
<td>77</td>
<td>240</td>
<td>*78</td>
<td>*2,540</td>
</tr>
<tr>
<td>ecos-icse11.dimacs</td>
<td>1,244</td>
<td>2,768</td>
<td>12</td>
<td>6</td>
<td>2</td>
<td>63</td>
<td>185</td>
<td>*64</td>
<td>*973</td>
</tr>
<tr>
<td>Eshop-fm.xml</td>
<td>287</td>
<td>22</td>
<td>5</td>
<td>3</td>
<td>0.16</td>
<td>21</td>
<td>5</td>
<td>108</td>
<td>457</td>
</tr>
</tbody>
</table>
Summary

- Our contribution
  - A scalable algorithm for t-wise (1-3) covering array generation
  - An empirical evaluation and comparison

- Implementation available
  - The implementation is available as open source (EPL)
  - Experiments are reproducible

- All the data is available
  - All 28,500 measurements available for the paper's resource website + charts and summaries