COMBINATORIAL TESTING FOR FEATURE MODELS USING CITLAB

Int. Workshop on Combinatorial Testing (IWCT) @ ICST 2013

Angelo Gargantini
Università di Bergamo - Italy

http://cs.unibg.it/gargantini

Joint work with Paolo Vavassori – Università di Bergamo
and Andrea Calvagna Università di Catania
SPLs, FM, and CIT

- **Software Product Lines & Feature Models**
  - SPLs and FMs are used to represent all the possible products of a software product line in terms of features and relationships among them.

- **Combinatorial Interaction Testing**
  - Often required for SPLs

- **Current approach**
  - Adapt CIT algorithms and tools for SPLs

- **OUR PROPOSAL   FM2CitLab**
  - Use a tool for combinatorial testing (CitLab) for test generation starting from Feature Models
Feature models

- In software product line engineering, feature models represent all possible products of a software product line in terms of features and relationships among them.

- Example
Feature IDE

http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

Diagram showing a feature model with nodes such as MobilePhone, Calls, GPS, Screen, Media, Basic, Color, High_Resolution, Camera, and Mp3, with relationships like Camera ⇒ High_Resolution and Basic ⇔ ¬GPS.
Features relationships in FMs

**MANDATORY**
- child feature A is mandatory
- Exactly one of the sub-features must be selected

**ALTERNATIVE**
- Exactly one of the sub-features must be selected

**OPTIONAL**
- At least one of the sub-features must be selected
- OR

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
Standard semantics

• Feature models semantics can be rather simply expressed by using propositional logics

• Every feature is translated to a **Boolean** input

+ Add constraints for the relations among features (implicit constraints)
  - Alternative features are expressed as exclusive or

+ Add constraints for cross-tree requirements
Disadvantages

10 Boolean variables Model, A, B, $a_1, ..., a_4, b_1, ..., b_4$

Constraints:
e.g. A is alternative:

For $A$:

$(a_1 \land \neg a_2 \land \cdots \land \neg a_4)
\lor
(\neg a_1 \land a_2 \land \cdots \land \neg a_4)
\cdots
\lor
(\neg a_1 \land \neg a_2 \land \cdots \land a_4)$
A “better” way to translate FMs to combinatorial problems

The translation to CitLab language is performed in the following steps

1. Every feature, starting from the root feature, is translated to an element (variable or literal constant) in the combinatorial problem.
   • Initialize also a function isChosen to be used when formalizing the constraints
2. Additional constraints are added in order to represent relationships among features as specified by the hierarchies in the future model.
3. Cross-tree constraints are translated and added to the model.
4. Apply some simplification
1. Parameters

Alternative

`Enumerative A {a1 ... an NONE};`

`isChosen(A) ≡ A != NONE`

`isChosen(ai) ≡ A == ai`

Everything else

`Boolean A;`

`isChosen(A) ≡ A == true`
2. Implicit constraints

For alternative no implicit constraints (unless... )

\[
isChosen(\{ai\}) \Rightarrow isChosen(A)
\]

\[
isChosen(p) \iff isChosen(A)
\]

For alternative no implicit constraints (unless... )
4. Simplification

- After translation, we simplify the model:
  1. Simplify the constraints in a semantic preserving way (equivalence)
  2. Remove unnecessary parameters and constraints.

- The resulting model is equisatisifiable as the original one
  - They allow the «same» family of products
  - Since some features are missing, products of the simplified model are more abstract.

It can be applied to any model, not only those coming from FMs

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
1. Constraints Simplification

<table>
<thead>
<tr>
<th>Constraint</th>
<th>If already present</th>
<th>Replaced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \Rightarrow b$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$a \Rightarrow b$</td>
<td>$b$</td>
<td>- remove</td>
</tr>
<tr>
<td>$a \Leftrightarrow b$</td>
<td>$a$</td>
<td>$b$</td>
</tr>
<tr>
<td>$a \Leftrightarrow b$</td>
<td>$b$</td>
<td>$a$</td>
</tr>
</tbody>
</table>

- In terms of FMs:

![Feature Model Diagram]

Legend:
- Mandatory
- Optional
- Abstract
- Concrete
- False optional

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
2. Parameter removal

<table>
<thead>
<tr>
<th>Parameter</th>
<th>If present</th>
<th>action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean ( A; )</td>
<td>( A == \text{true} )</td>
<td>Remove ( A ) and the constraint</td>
</tr>
<tr>
<td></td>
<td>( A == \text{false} )</td>
<td>Remove ( A ) and the constraint</td>
</tr>
<tr>
<td>Enumerative ( A {a_1 \ldots a_n}; )</td>
<td>( A == a_1 )</td>
<td>Remove ( A ) and the constraint</td>
</tr>
<tr>
<td></td>
<td>( A != a_1 )</td>
<td>Remove ( a_i ) and the constraint</td>
</tr>
</tbody>
</table>

- In terms of FM
d- Some features:
  - *display, backCamera, Phone*

  are always present, can be ignored

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
Experiments

• Over 52 feature models from SPLOT repository
• Using the FeatureIde parser
  • We had to skip some models because of its faults
• Implemented the BOOL translation for comparison

1. Testing the correctness of the transformation
  • We have not proved that our translation is correct, but tested against the BOOL (by the number of valid products)

2. Effect of the simplification over the parameters and the constraints

3. Comparison with BOOL in terms of model
  1. # parameters: we should obtain smaller models
  2. # constraints: we should obtain simpler models
  3. variability: we should obtain more compact models

4. Test generation vs BOOL
2. Simplification effect

No reduction

Ratio: \( \frac{x \text{ after}}{x \text{ before}} \)

Reduction to 0

Average:
- 77% of parameters
- 43% number of constraints
- 38% number of clauses in them

In 11% of cases complete reduction of constraints

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
3 Vs. BOOL

Reduction of:
- # parameters
- # CNF clauses in constraints

Increase of variability

\[ \text{var} = \frac{\text{# valid products}}{\text{# all products}} \]

Ratio: \( \frac{x \text{ after}}{x \text{ before}} \)

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
4. Test generation time (vs. BOOL)

- Can test generators take advantage of our translation?
- vs BOOL (+ simpl) using

<table>
<thead>
<tr>
<th>BOOL + cit</th>
<th>FM2CitLab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>Size</td>
</tr>
<tr>
<td>0.785</td>
<td>39</td>
</tr>
</tbody>
</table>

Often we had fewer test cases BUT more time

Angelo Gargantini - Combinatorial Testing for Feature Models Using CitLab
Other results

- Tools for combinatorial testing performed much better than tools specifically developed for SPLs (PACOGEN, OSTER)
Conclusions

• A new **better** way to translate FMs to combinatorial problems
  • More compact
  • Fewer parameters and constraints
  • Increased variability
• Integrated into CitLab
  • Reuse of test generators

THANK YOU