An ongoing work on Statistical Structural Testing via Probabilistic Concurrent Constraint Programming

Matthieu Petit¹ and Arnaud Gotlieb

¹: Projet GENETTA (GENERation automatique de Tests sTastiques pour jAva embarqué)
Statistical Testing

- **Random** test data generation

- Existence of a probabilistic model to estimate the quality of the test coverage

**Advantages:**
- Randomness in the test data selection
- Automated method

**Drawbacks:**
- No guarantee of the test criterion coverage
- Automated oracle is required
Statistical Testing as model-based testing

- Description of the usage of the software under test (operational profiles, input distributions)

- Representation with a probabilistic system of transitions (Markov Chain, Flow Graphs)
  [Whittaker et al. 94, Walton et al. 95, Thevenod and Waeselynck 89]
Our approach

- Statistical testing for the coverage of **structural** testing criteria
- Build a model from a control flow analysis and a computation of theoretical probabilities
- Deriving an input distribution from **probabilistic constraint solving**
Outline

- Motivating example
- Statistical structural testing as model-based testing
- Probabilistic Concurrent Constraint Programming over Finite Domain (PCCP(FD))
- Statistical Structural Testing as PCCP problem
Example

 Representation of the program by its control flow graph (CFG)

if \(x \leq 100 \land y \leq 100\)

if \(y > x + 50\)

if \(y \times x < 100\)
Random Testing

- **Uniform** probability distribution over the input domain

- Problem: certain elements of the program have a very low probability to be activated

- only 100 input points over $2^{64}$ activate the path 1-2-3-4-5-6-7
Statistical Structural Testing
[Thevenod, Waeselynck 89]

- Non-uniform probability distribution over the input domain
- Purpose: maximize the probability to activate any element of the program
- Example (All-Paths): each path must be activated with the probability 1/5
Outline

- Motivating example

- Statistical structural testing as model-based testing

- Probabilistic Concurrent Constraint Programming over Finite Domain (PCCP(FD))

- Statistical Structural Testing as PCCP problem
A possible model for the statistical structural testing

- Notations:
  - \( C \): testing criterion
  - \( S_C \): set of the elements to activate
  - \( p_C = \min_{k \in S_C} (p(k)) \)

- Extension of CFG with edge labels
  \[ L_C = \{ p(e) | e \in E : \text{set of the CFG edges} \} \]
  such as \( p_C \) is maximized
Example Model

\( L_{\text{All-Nodes}} \)

\( L_{\text{All-Edges}} \)

\( p_C = \frac{1}{1} \)

\( p_C = \frac{1}{3} \)
Example Model

\[ L_{\text{All-Paths}} \]

1 — 2 — 3 — 4 — 5 — 6 — 7

\[ p_C = \frac{1}{5} \]
Purpose of the random test data generation

Criterion: All-Paths

1. if $(x < 100 \text{ and } y < 100)$
2. if $(y > x + 50)$
3. if $(y \times x < 100)$
4. if $(y \times x < 100)$
5. if $(y \times x < 100)$
6. if $(y \times x < 100)$
7. ush x ush y

\[
\begin{align*}
p((x \leq 100 \land y \leq 100) \land (y > x + 50) \land (y \times x < 100)) &= \frac{1}{5} \\
p((x \leq 100 \land y \leq 100) \land (y \leq x + 50) \land (y \times x < 100)) &= \frac{1}{5} \\
p((x \leq 100 \land y \leq 100) \land (y > x + 50) \land (y \times x \geq 100)) &= \frac{1}{5} \\
p((x \leq 100 \land y \leq 100) \land (y \leq x + 50) \land (y \times x \geq 100)) &= \frac{1}{5} \\
p(x > 100 \lor y > 100) &= \frac{1}{5}
\end{align*}
\]
Outline

- Motivating Example

- Statistical structural testing as model-based testing

- Probabilistic Concurrent Constraint Programming over Finite Domain (PCCP(FD))

- Statistical Structural Testing as PCCP problem
Probabilistic Concurrent Constraint Programming (PCCP)

[Gupta et al. 97, Di Pierro et al. 98]

- Constraint Store: conjunction of constraints on the possible values of variables

- Process
  - Syntax:
    - $P ::= \text{tell}(C) \mid \text{if } C \text{ then } P \mid P \parallel P$  \hspace{1cm} (CCP)
  - Probabilistic choice operator:
    - $\text{choose } X \text{ from } \text{Dom} \text{ with distribution } f \text{ in } P$
Example

choose $X$ from $(0,1)$ with distribution $(1/2, 1/2)$ in $\{\text{tell}(X=Z)\} \parallel$
choose $Y$ from $(0,1)$ with distribution $(1/2, 1/2)$ in $\{\text{if } Z=1 \text{ then tell}(Y=1)\}$
Example

choose $X$ from $(0,1)$ with distribution $(1/2,1/2)$ in $\{\text{tell}(X=Z)\} \parallel$
choose $Y$ from $(0,1)$ with distribution $(1/2,1/2)$ in $\{\text{if } Z=1 \text{ then } \text{tell}(Y=1)\}$

\[
\begin{align*}
p(X = 0) &= \frac{1}{2} \\
p(X = 1) &= \frac{1}{2} \\
p(X = 1 \land Y = 1) &= \frac{1}{4} \\
p(X = 1 \land Y = 0) &= \frac{1}{4}
\end{align*}
\]
Implementation of the probabilistic choice operator

- Use of a constraint solver over finite domains (library clp(FD) of SICStus Prolog)

- Originality: the operator is a **new global constraint** of the constraint solver
  - the probabilistic choice can accept unbound parameter (domain, distribution probability)

choose(X, [V_1, V_2, ..., V_n], [W_1, W_2, ..., W_n], Process)
How does it work?

- The random drawing is not feasible while the domain or the probability distribution is unknown.

- But, we can exploit partial information to prune the domains of the possible values.

- The random drawing is delayed until complete information is available.
New probabilistic choice operator: choose_dec/5

- Simulate the probabilistic behavior of the conditional statement in clp(FD)

- Syntax:

  ```
  choose_dec(Constraint, W1, W2, Process1, Process2)
  ```

choose_dec/5 is true iff (Constraint \land Process1) is true with a probability \(\frac{W_1}{W_1 + W_2}\) or (\neg Constraint \land Process2) is true with a probability \(\frac{W_2}{W_1 + W_2}\)
Outline

- Motivating Example
- Statistical structural testing as model-based testing
- Probabilistic Concurrent Constraint Programming over Finite Domain (PCCP(FD))
- Statistical Structural Testing as PCCP problem
Motivating example as a PCCP problem

\[
\text{if } (x<100 \land y<100) \quad \text{X in 0..1000, Y in 0..1000},
\]

\[
\text{if } (y>x+50)
\]

\[
\text{if } (y\times x<100)
\]
Motivating example as a PCCP problem

if \((x <= 100 \&\& y <= 100)\)

if \((y > x + 50)\)

if \((y \cdot x < 100)\)

\(\text{ush } x \text{ ush } y\)

\(X \text{ in } 0 \ldots 1000, \ Y \text{ in } 0 \ldots 1000,\)

[\text{choose\_dec}(Y > X + 50, W1, W2, [], [])]
Motivating example as a PCCP problem

if (x<=100 && y<=100)

if (y>x+50)

if (y*x<100)

choose_dec(Y>X+50, W1, W2, [], []),
choose_dec(X*Y<100, W3, W4, [], [])
Motivating example as a PCCP problem

X in 0..1000, Y in 0..1000,
.choose_dec(X≤100∧Y≤100,W5,W6,
[choose_dec(Y>X+50),W1,W2,[]],[[]]),
.choose_dec(X*Y<100,W3,W4,[]],[[]]),[],[])

ush x ush y

if (x=<100 && y=<100)

if (y>x+50)

if (y*x<100)
Motivating example as a PCCP problem

Criterion: All-Paths

ush x ush y

if (x=<100 && y=<100)

if (y>x+50)

if (y*x<100)

X in 0..1000, Y in 0..1000,
choose_dec(X<=100 && Y<=100, W5, W6,
[choose_dec(Y>X+50, W1, W2, [], []),
choose_dec(X*Y<100, W3, W4, [], [])], []),
W1=1, W2=1, W3=1, W4=1, W5=4, W6=1.
**Motivating example as a PCCP problem**

Criterion: All-Paths

- Any path is activated with a probability $\frac{1}{5}$

Different constraint stores after 5000 process executions:

- $[(X \in 0..100, Y \in 0..100), 0.2126] % \text{path 1-2-3-4-5-6-7}$
- $[(X \in 1..49, Y \in 52..100), 0.1956] % \text{path 1-2-3-4-6-7}$
- $[(X \in 0..100, Y \in 0..100), 0.2004] % \text{path 1-3-4-5-6-7}$
- $[(X \in 1..100, Y \in 1..100), 0.1954] % \text{path 1-3-4-6-7}$
- $[(X \in 0..1000, Y \in 0..1000), 0.1960] % \text{path 1-7}$

Diagram:

1. **ush x ush y**

2. if $(x < 100 \land y < 100)$

3. if $(y > x + 50)$

4. if $(y \times x < 100)$

5. X in 0..1000, Y in 0..1000,

6. choose_dec($X \leq 100 \land Y \leq 100, W_5, W_6,$

7. [choose_dec($Y > X + 50, W_1, W_2, [], []),

8. choose_dec($X \times Y < 100, W_3, W_4, [], []), [], []], [W_1 = 1, W_2 = 1, W_3 = 1, W_4 = 1, W_5 = 4, W_6 = 1$.

1011/04
Random data test generation

Random data test generation are solutions of constraint system

Two processes are used during the constraint solving:
- the constraint propagation
- a random labelling: process that selects a tuple of values for input variables at random
Ongoing Work

- Extend the translation of an imperative program into PCCP problem for a representative language:
  - Efficient treatment of the loop statement [Gotlieb et al. 00]
  - Treatment of method calls
Ongoing work

- Extension of our model to take account of dynamic information
  - Exploiting the detection of (some) non-feasible paths

- Edge labels as variables which are constrained during the random test data generation

- Application to real world program testing (Java Card, J2ME)