Debugging of Model Transformations and Contracts in SyVOLT

Bentley James Oakes, Clark Verbrugge, Levi Lúcio, Hans Vangheluwe

McGill University, fortiss GmbH, University of Antwerp, Flanders Make

October 16, 2018
1. Verification activity
   Proving structural contracts

2. Debugging
   Detecting/localizing artefact errors in the verif. activity

Experience report

Debugging in Verif. Tool - Verification vs. Debugging - Debugging Improvements
1 Verification Activity

2 Debugging Stage 1: Analysis

3 Debugging Stage 2: Monitoring

4 Debugging Stage 3: Reporting

5 Conclusion
Motivation

- GIVEN: A transformation divided into layers, containing LHS/RHS rules
- GOAL/WHY: Understand transformation’s behaviour
  - Relation between input/output elements

- WHAT: Prove structural contracts to guarantee element existence
- HOW: Create all possible rule combinations through symbolic execution

- Rules are arranged in layers, where each layer fully executes before the next
- Rules have *Match* part and *Apply* part
- Reduced expressiveness - no deletion/loops
Symbolic Execution

Goal: Create all possible transformation executions

Example: Combine four rules into a path condition:

- Symbolically execute each layer of the transformation
- Resolve dependencies between rules
- Final set of path conditions represents all valid transformation possibilities
Contract Proving

Contract: “A Family with a daughter and a mother always produces a Man element”

Contract elements matched onto path condition

Matching failure indicates **counter-example** to the contract
  - Set of rules as counter-example
SyVOLT Tool

Debugging Stage 1: Analysis

Contracts

Debugging Stage 2: Monitoring

Transformation

Verification Process

Success

Debugging Stage 3: Reporting

Counter-example rule combinations
Outline

1 Verification Activity

2 Debugging Stage 1: Analysis

3 Debugging Stage 2: Monitoring

4 Debugging Stage 3: Reporting

5 Conclusion
Before symbolic execution, **analyze** transformation and contracts

- Sanity check - transformation/contract valid
- Record-keeping - record dependencies

“A Family with a father, mother, son and daughter should always produce two Man and two Woman elements connected to a Community”

- Are contract elements present in the transformation?
- Are element creation dependencies satisfied?
- Which rules does this contract depend on?
  - Enables **slicing** - selecting subset of rules to symbolically execute
Fixing Input Errors

Rule:

- Woman in rule $\neq$ Female in contract
- Typos/inconsistencies prevent satisfying contracts

Analysis:
- Check if elements and dependencies are satisfied
  - Error: Meta-model element ‘Female’ not found in any rule!
- Lists of rules this contract depends on
  - Required rules for contract Pos_FourMembers:
    - [‘Daughter2Woman’, ‘Father2Man’, ‘Mother2Woman’, ‘Son2Man’…]
Reducing Errors

- Contract/rule elements must be typed by transformation meta-models
  - Should be enforced by tooling

MPS:

```
Rule Name: Father2Man
Match Model
Classes
2.0.m.0Parent : concept/Parent/ Any MatchClass (Allow inheritance = false )

firstName : property/Parent : firstName/ : String Matcher>>
2.0.m.1Family : concept/Family/ Exists MatchClass (Allow inheritance = false )

lastName : property/Family : String Matcher>>

Links
2.0.m.0Parent ---- Direct Match
2.0.m.1Family ---- Direct Match
Apply Model
Classes
2.0.a.0Man : concept/Man/ ApplyClass (Allow inheritance = false )

fullName : property/Man : fullName/ : (Ref. to) firstName + (Ref. to) lastName

Backward Links
<< ... >>
```

Discussion Question: Bug prevention is not debugging, but highly related
- Debugging can be generalizing larger classes of bugs?
Outline

1 Verification Activity

2 Debugging Stage 1: Analysis

3 Debugging Stage 2: Monitoring

4 Debugging Stage 3: Reporting

5 Conclusion
Stage 2: Monitoring

- Recall: SyVOLT performs symbolic execution before proving contracts
- **Monitor** that all rules are symbolically executed

**Symbolic Execution Tree:**

Error: Rule ‘A’ was not symbolically executed on layer C!
Rule ‘A’ depends on rules: [...]  

**Causes:**
- Multiplicity issue where dependency is not executed enough times
- Technique to remove invalid path conditions  
  - Invalid means not respecting containment constraints
1 Verification Activity

2 Debugging Stage 1: Analysis

3 Debugging Stage 2: Monitoring

4 Debugging Stage 3: Reporting

5 Conclusion
Stage 3: Reporting

- Verification produces counter-examples (rule combinations) to a contract
- Want to report why a particular contract is not satisfied

a) Name: Neg_SchoolOrdFac
   Num Succeeded Path Conditions: 6
   Num Failed Path Conditions: 3

b) Explaining contract result:
   Good rules: (Rules in success set and not failure set)
   \textit{dfacilities}...OrdinaryFacilityPerson
   Bad rules: (Rules common to all in failure set)
   \textit{dfacilities}...SpecialFacilityPerson

c) Contract requires elements from successful rules of type:
   School
   OrdinaryFacility

Discussion Question: Is this output \textit{debugging} or \textit{verification}?
Counter-example to the `Neg_SchoolOrdFac` contract has a `SpecialFacility` instead of an `OrdinaryFacility`.

Better visualization required!

What elements make the contract succeed?

If the contract fails, what changes would make the contract succeed?
1 Verification Activity

2 Debugging Stage 1: Analysis

3 Debugging Stage 2: Monitoring

4 Debugging Stage 3: Reporting

5 Conclusion
SyVOLT verification tool performs debugging of transformation and contracts in three stages:

- **Stage 1:** *Analysis* - dependency information
- **Stage 2:** *Monitoring* - ensure correct symbolic execution
- **Stage 3:** *Reporting* - relate contract failure to involved elements

**Discussion Questions:**

- Line between verification and debugging?
- Is debugging = observation of behaviour?
- How does prevention of errors relate to debugging?
- Improvements for debugging visualization?
  - For verification itself, and development of the verification tool

Thank you!

*Debugging of Model Transformations and Contracts in SyVOLT*

**Bentley James Oakes**, Clark Verbrugge, Levi Lúcio, Hans Vangheluwe