Optimized Translation of Clafer Models to Alloy

Kacper Bak
Generative Software Development Lab
University of Waterloo

CS744 Course Project. July 19, 2011

Course Project

CS 744: Advanced Compiler Design

Data flow analysis, redundancy elimination, optimizations

Individual project

Duration: 2 months

Clafer Update

Analysis of variability models

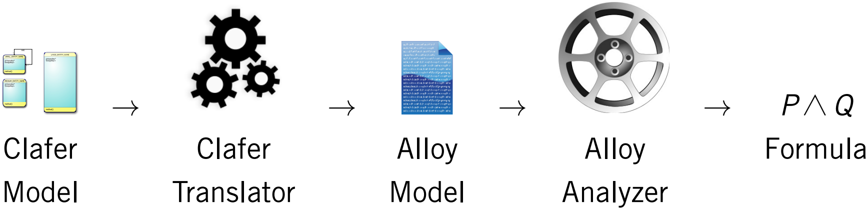
Translation to Alloy (uses SAT solvers)

clafer2alloy translator: a year ago

Some work on formal semantics

Examples of variability models

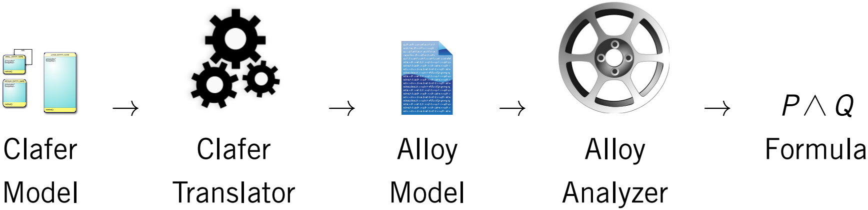
The Toolchain



Clafer Model → Clafer Translator → Alloy Model → Alloy Analyzer → $P \wedge Q$ Formula

Demo

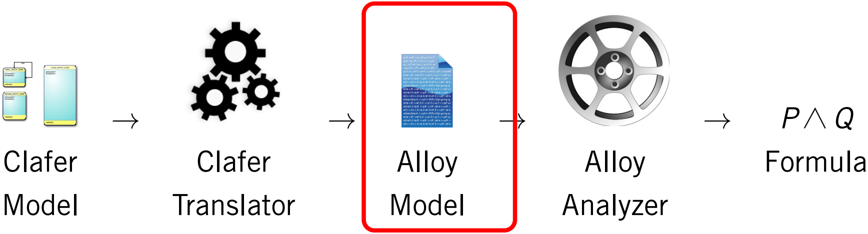
Problems



Clafer Model → Clafer Translator → Alloy Model → Alloy Analyzer → $P \wedge Q$ Formula

Translation rules heavily influence reasoning time in Alloy

Problems

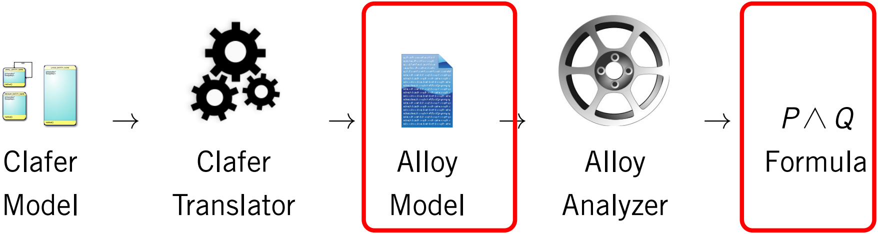


Clafer Model → Clafer Translator → Alloy Model → Alloy Analyzer → $P \wedge Q$ Formula

Translation rules heavily influence reasoning time in Alloy

Large Alloy files (complex models)

Problems



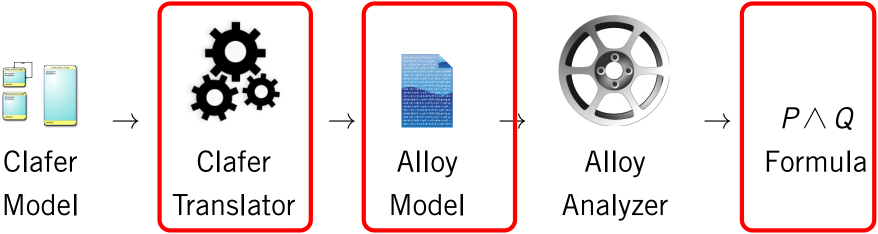
Clafer Model → Clafer Translator → Alloy Model → Alloy Analyzer → $P \wedge Q$ Formula

Translation rules heavily influence reasoning time in Alloy

Large Alloy files (complex models)

Ineffective Alloy representation (complex formulas)

Problems



Clafer Model → Clafer Translator → Alloy Model → Alloy Analyzer → $P \wedge Q$ Formula

Translation rules heavily influence reasoning time in Alloy

Large Alloy files (complex models)

Ineffective Alloy representation (complex formulas)

Slow clafer2alloy translator

Solution

Refactored and modular code architecture

User has control over the translation process

Intermediate language representation

Optimization of translation rules

The Translator

(Old) clafer2alloy Translator

Parser, desugarer, semantic analyzer, code generator

Monolithic

Haskell

Available online

Released source code

(New) clafer Translator

Front-end, intermediate representation, optimizer, generators

User can turn on/off modules (has extra knowledge)

Easy to add new code generators

Optimizations

No Unused Abstract Clafers

```
abstract display
  server ?

OnBoardComputer
```

OnBoardComputer

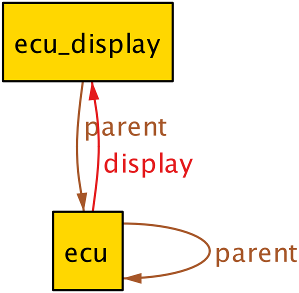
No Unused Abstract Clafers

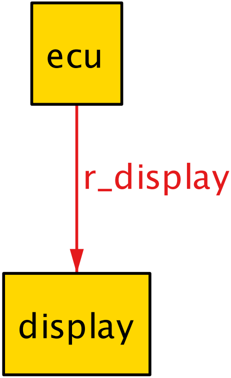
```
abstract display
  server ?

OnBoardComputer
```

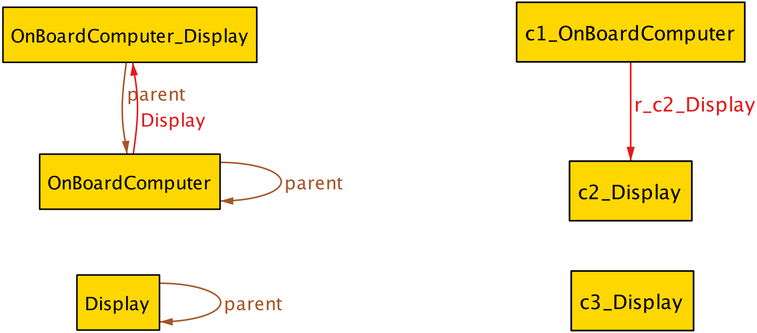
```
OnBoardComputer
```

No Redundant Hierarchical Constraints





Improved Name Resolution



Global Cardinality Constraints

```
OnBoardComputer 0..1
  Display 1
```

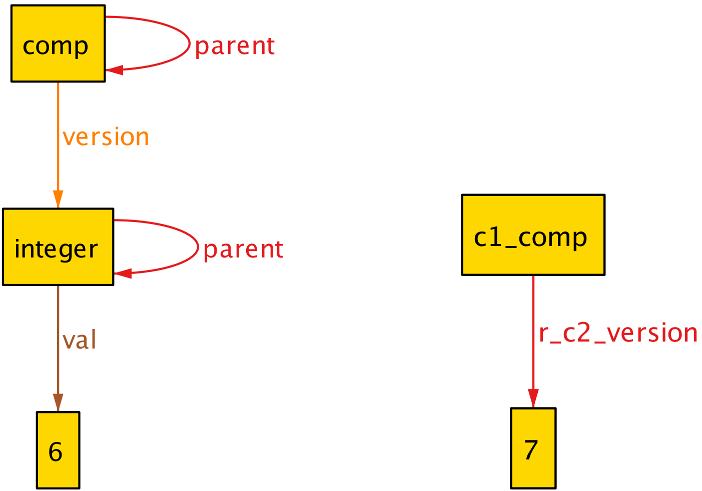
```
OnBoardComputer 0..1
  Display 0..1
```

Global Cardinality Constraints

```
OnBoardComputer 0..1
  Display 1
```

```
OnBoardComputer 0..1
  Display 0..1
```

Integers as Attributes



References are Relations

```
ecu

display
  server -> ecu
```

```
sig display extends clafer
{ server : one clafer }
{ server in ecu }
```

```
one sig display
{ server : one ecu }
```

References are Relations

```
ecu

display
  server -> ecu
```

```
sig display extends clafer
{ server : one clafer }
{ server in ecu }
```

```
one sig display
{ server : one ecu }
```

References are Relations

```
ecu

display
  server -> ecu
```

```
sig display extends clafer
{ server : one clafer }
{ server in ecu }
```

```
one sig display
{ server : one ecu }
```

Unrolled Inheritance

```
abstract comp
  version : integer

display extends comp
```

```
display
  version : integer
```

Unrolled Inheritance

```
abstract comp
  version : integer

display extends comp
```

```
display
  version : integer
```

Model Statistics

```
ecu 1..2
  display -> integer 2..3
  [display > 2]
```

```
All clafers: 2 | Abstract: 0 | Concrete: 1 | References: 1
Constraints: 1
Global scope: 1..3
All names unique: False
```

Model Statistics

```
ecu 1..2
  display -> integer 2..3
  [display > 2]
```

```
All clafers: 2 | Abstract: 0 | Concrete: 1 | References: 1
Constraints: 1
Global scope: 1..3
All names unique: False
```

Parameters

Unrolling inheritance

Timeout for model translation

Layout resolver options

Checking duplicated names

Name resolver behavior

Keeping unused clafers

Evaluation

Input Models

Feature Models (instantiation)

Meta-Models (instantiation)

FBMTs (liveness, instantiation)

The Linux Kernel

Results

Speed: 2-5 times faster

Possible to handle huge models

Conclusion

Conclusion

Clafer models can be expressive and analyzable

Possible further optimizations

User knowledge is very useful

Thanks for listening!

Questions?

gsd.uwaterloo.ca/clafer