EE382C-3 Verification and Validation

khurshid@ece.utexas.edu

Lecture 2

January 16, 2025

Today

Alloy [http://alloytools.org/]

Next week

Specification-based testing

- TestEra [ASE’01, ASE-J’04]
- Korat [ISSTA’02]

Alloy: Overview

"Analyzable models for software design"

Declarative language

- First-order logic with transitive closure
- Based on relations

Two design goals

- Simple language for modeling
- Automatic analysis

Developed by Daniel Jackson and his group at MIT

Alloy tool-set

Alloy language – build models, requirements, specifications, design

Alloy analyzer – automatic analysis tool

Why do we need such a tool?

Alloy language

- Provides precise description of artifacts
- Documentation
- Provides higher level of abstraction
- Helps describe properties that we cannot (easily) express in source code

Alloy analyzer

- Enables machine reasoning
- Helps eliminate/reduce ambiguities, inconsistencies, and incompleteness

Informal statement example

"Everybody likes a winner"

- Ambiguous?
- Incomplete?

Precise meaning?

- all p: Person | some w: Winner | p.likes(w)
- all p: Person | all w: Winner | p.likes(w)
- some w: Winner | all p: Person | p.likes(w)
- some w: Winner | some p: Person | p.likes(w)

Alloy: Basic concepts

Atom (or scalar)

Set

Relation

Universe of discourse

Quantification

- Universal
- Existential

Modeling structures

Singly-linked list

```
module list

sig List { header: lone Node }

sig Node { next: lone Node }

pred RepOk(l: List) { all n: l.header.*next | n not in n.^next }
run RepOk for 3
```

Checking example

```
module list

sig List { header: lone Node }

sig Node { next: lone Node }

pred RepOk(l: List) { all n: l.header.*next | n not in n.^next }

pred RepOk2(l: List) {
  no l.header || some n: l.header.*next | no n.next
}

assert Equiv { all l: List | RepOk[l] <=> RepOk2[l] }
check Equiv for 3
```

Another example

Russell's paradox: *in a village where the barber shaves every man who doesn't shave himself, who shaves the barber*?

```
module russell

sig Man {
  shaves: set Man } // shaves: Man x Man

one sig Barber extends Man {}

pred Paradox() {
  Barber.shaves = { m: Man | m not in m.shaves } }
run Paradox for 4
```

Where’s the paradox? [Dijkstra]

Russell’s paradox: *in a village where the barber shaves every man who doesn’t shave himself, who shaves the barber*?

```
module russell

sig Man {
  shaves: set Man } // shaves: Man x Man

/* one */ sig Barber extends Man {}

pred Paradox() {
  Barber.shaves = { m: Man | m not in m.shaves } }
run Paradox for 4
```

Everything is a set

No scalars

Alloy equates the following

- a (atom)
- <a> (one-tuple)
- {a} (singleton set)
- {<a>} (singleton set of a one-tuple)

Makes navigation easier

Relational composition “x.y”

Let

- x be a relation of type T1->T2->…->Tm
- y be a relation of type S1->S2->…->Sn
- m+n > 2

[[x.y]] = { <t1…t(m-1) s2…sn> |
some c. <t1…t(m-1) c> in [[x]] /\
<c s2…sn> in [[y]] }

Common case – “x” set, “y” binary relation: “x.y” is *relational image*

“x” and “y” binary relations: “x.y” is *relational composition*

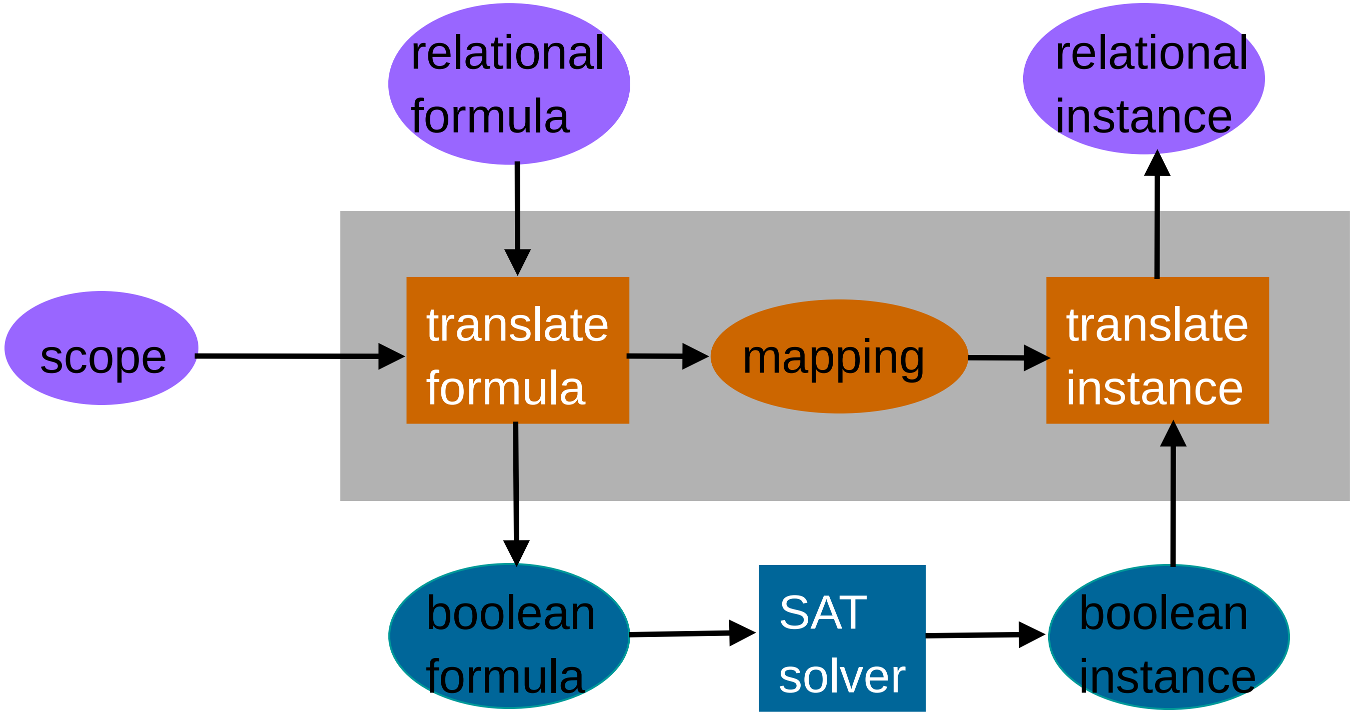
Alloy analyzer (back-end)

Translates to boolean formulas, employs SAT

Provides two types of analysis – technically the same

- Simulation finds instances
- Checking finds counterexamples

Alloy analyzer architecture



Example formula

Alloy formula

```
all y: Y | X.r != y
```

Y and r are “global variables”

```
sig Y {}
sig X {
  r: Y
}
```

Scope of 2

$|X| = |Y| = 2$

Example representation

X = [x0 x1]

element is either in the set or not

$$r = \begin{vmatrix} r00 & r01 \\ r10 & r11 \end{vmatrix}$$

$$x.r = [x0\ x1]\ o \begin{vmatrix} r00 & r01 \\ r10 & r11 \end{vmatrix} = [x0\ r00 \lor x1\ r10 \quad x0\ r01 \lor x1\ r11]$$

Alloy analyzer discussion

How well does it scale?

What kind of bugs can it find?

When does “small scope hypothesis” hold?

Usage in other tools

- Software architecture descriptions
- Requirements analysis
- Refinement tool
- ...

Research topics (1)

Improve analysis

- Compiler optimizations
- Incremental analysis
- SAT optimizations
- Symmetry breaking

Try new case studies

- Can you model something from your domain?
- Can you analyze the model?

Research topics (2)

Invent novel applications

- Analysis of code, specs etc.
- Synthesis of code, specs etc.
- ML

Re-design the language

- What to add/remove, but to remain analyzable?

Change the analysis

- SMT instead of SAT

Some applications of Alloy

Software testing (TestEra – next class)

- Automatic generation of test inputs

Static analysis (JAlloy)

- (Unsound) checking of code conformance

Case studies

- Networking protocols (INS, Chord)
- Software models (COM)
- Distributed algorithms

Synthesis of chemical reaction networks, neural networks etc.

Analysis of ML models

?/!