An Overview of the Alloy Language & Analyzer

Slides contain some modified content from the Alloy Tutorial by G. Dennis & R. Seater

(see alloy.mit.edu)

What is Alloy?

- A formal language and analyzer based on Z
- Developed at MIT by Daniel Jackson and his team
- Based on relations, where a relation is a set of tuples
  - A tuple is a sequence of atomic items
- Treating all entities as relationships makes it easier to analyze Alloy models

Understanding Alloy

- Three parts
  - The logic
    - First-order expressions on relations
    - Relations of relations (i.e., higher-order relations) are not supported
    - States and executions are described using constraints (like Z, OCL)
  - The language
    - Provides structure and "syntactic sugar"
  - The analysis mechanism
    - Takes the form of constraint solving
    - Simulation: Find instances that satisfy a set of constraints
    - Checking: Find a counterexample that violates a constraint

Structure of an Alloy Model

```
module tour/addressBook1h ------- Page 14..16

sig Name, Addr { }

sig Book {
    addr: Name -> lone Addr
}

pred show [b: Book] {
    #b.addr > 1
    #Name.(b.addr) > 1
}
run show for 3 but 1 Book

pred add [b, b': Book, n: Name, a: Addr] {
    b'.addr = b.addr + n->a
}

pred del [b, b': Book, n: Name] {
    b'.addr = b.addr - n->Addr
}

assert delUndoesAdd {
    all b, b', b'': Book, n: Name, a: Addr |
        no n.(b.addr) and add [b, b', n, a] and del [b', b'', n]
        implies
        b.addr = b''.addr
}

// This command should not find any counterexample.
check delUndoesAdd for 3
```

Module header

Signatures: A signature declares a set of atoms
- **can also introduce field**
- **each field represents a relation**

commands are in red

Constraint paragraphs: specifies constrainst (e.g., invariants)

Assertions: properties that are expected to hold

A world of relations …

Everything is a relation in Alloy

– A relation is a set of tuples

- sets are unary (1 column) relations

Name = {(N0), (N1), (N2)}
Addr = {(A0), (A1), (A2)}
Book = {(B0),(B1)}

- scalars are singleton sets

myName = {(N1)}
yourName = {(N2)}
myBook = {(B0)}

- binary relation

names = {(B0, N0),
(B0, N1),
(B1, N2)}

- ternary relation

addrs = {(B0, N0, A0),
(B0, N1, A1),
(B1, N1, A2),
(B1, N2, A2)}

Analysis in Alloy

- Analysis: find some assignment of values (relations) to variables that makes a constraint true
- You can ask Alloy to perform 2 types of constraint/assertion checks
  - Find an instance of a model that satisfies constraints (use the **run** command)
  - Find an instance in which an assertion does not hold; the instance is called a counterexample (use the **check** command)
- Analysis is made tractable by restricting the space in which it searches for solutions
  - Defining the restricted search space is called *scope setting*

Alloy language elements: Signature Fields

- Signature field
  - A field in a signature is a relation in which the domain is a subset of the signature elements
- **sig** A {f: e}
  - *f is a binary relation with domain A and range given by expression e*
  - *f is constrained to be a function*
  - *(f: A -> e)*

Alloy language elements: Constraints

- A fact is a constraint that is intended to always hold
- An assertion is a constraint that is intended to follow from facts
- A predicate is a reusable constraints, i.e., it is used to express facts and assertions
- A function defines a reusable expression

Alloy language elements: the run command

pred p[x: X, y: Y, ...] { F }
run p *scope*

- *instructs analyzer to search for instance of predicate within scope*

```
pred show [b: Book] {
  #b.addr > 1
  #Name.(b.addr) > 1
}
run show for 3 but 1 Book
```


Example (from tutorial)

sig Platform {}
there are “Platform” things

sig Man {ceiling, floor: Platform}
each Man has a ceiling and a floor Platform

pred Above[m, n: Man] {m.floor = n.ceiling}
Man m is “above” Man n if m's floor is n's ceiling

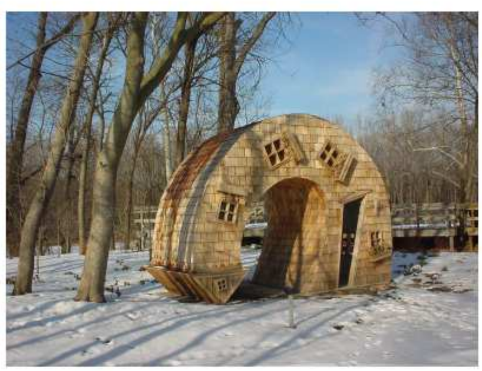
fact {all m: Man | some n: Man | Above (n,m)}
"One Man's Ceiling Is Another Man's Floor"

assert BelowToo {
all m: Man | some n: Man | Above (m,n)
}
"One Man's Floor Is Another Man's Ceiling"?

check BelowToo for 2
check "One Man's Floor Is Another Man's Ceiling" counterexample with 2 or less platforms and men?

– counterexample found

A counterexample (from MIT Alloy tutorial)



McNaughton