Lecture 4: Introduction to Alloy
Outline

- Introduction to basic Alloy constructs using a simple example of a static model
  - How to define domains, subsets, relations with multiplicity constraints
  - How to use Alloy’s quantifiers and predicate forms
- Basic use of the Alloy
  - Loading, compiling, and analyzing a simple Alloy specification
  - Adjusting basic tool parameters
  - Using the visualization tool to view instances of models
Example: Family Structure

- We want to...
  - Model parent/child relationships as primitive relations
  - Model spousal relationships as primitive relations
  - Model relationships such as “sibling” as derived relations
  - Enforce certain biological constraints via 1st-order predicates (e.g., only one mother)
  - Enforce certain social constraints via 1st-order predicates (e.g., a wife isn’t a sibling)
  - Confirm or refute the existence of certain derived relationships (e.g., no one has a wife with whom he shares a parent)
Signatures

- Sets are introduced as signatures
  - sig A { } //A is a set
  - sig B in A { } //B is a subset of A
- Disjoint subsets
  - sig C, D extends A { } //C, D disjoint
- Partition
  - abstract sig A { }
- Typical Alloy modeling strategy
  - Identify basic domains, then declare other sets of interest as subsets
Example: Family Structure

**Alloy Model**

```alloy
module cs6610/Family
sig Person { }
sig Man extends Person { }
sig Woman extends Person { }
sig Married in Person { }
```

**Graphical Representation**

```
Married
  --

Person
  --

  Man
  --

Woman
```
Which of the following are instances of our current model?

A. Person = \{P0,P1,P2\}
   Man = \{P1,P2\}
   Married = {} 
   Woman = \{P0\}

B. Person = \{P0,P1,P2\}
   Man = \{P1,P2\}
   Married = {} 
   Woman = \{P0,P1\}

C. Person = \{P0,P1,P2,P3\}
   Man = \{P0,P1,P2,P3\}
   Married = \{P2,P3\}
   Woman = {} 

D. Person = \{P0,P1\}
   Man = \{P0\}
   Married = \{P1\}
   Woman = {} 

E. Person = \{P0,P1\}
   Man = \{P0\}
   Married = \{P1,P0\}
   Woman = \{P1\} 

The correct instances are: A, C, E.
Relations

- Declaring a relation between two sets $A$ and $B$
  
  \[
  r : A \rightarrow B \\
  r1, r2 : A \rightarrow B
  \]

- Comments on notation...
  
  - denotes $r$ is a subset of $A \times B$
  - Written with $\rightarrow$ because we will often think of $r$ as a “mapping” from $A$ to $B$... but $r$ is not necessarily a function
Example: Family Structure

**Alloy Model with siblings**

```alloy
model cs6610/Family
sig Person {
    // Relations
    siblings : set Person
}
sig Man, Woman extends Person{}
```

**Example instance**

```
Person = {P0,P1,P2,P3}
Man = {P1,P2}
Married = {}
Woman = {P0,P3}
Relations:
siblings = {P0 -> {P1,P2},
P1 -> {P0,P2},
P2 -> {P0,P1}}
```

Intuition: P0,P1,P2 are siblings
Multiplicities

Allow us to constrain the sizes of sets, including the definition domain and the image of a relation.

There are three multiplicities:
- some: one or more
- lone: zero or one
- one: exactly one

Examples:
- red: one Color // set red contains exactly one color
- favorite: lone Person // at most one favorite person
Multiplicities

- “one” is implicit in declaration of set variables
- Use “set” to override
  - siblings : set Person
Multiplicities and Relations

Multiplicities can be applied to the domain, range or both of a relation.

- **f : S -> lone T**
  - says that, for each element s of S, f maps s to at most a single value in T

- Potential instances:

  - s1 \rightarrow t1
  - s2 \rightarrow t2
  - s3 \rightarrow t3
  - s4 \rightarrow t4

  - s1 \rightarrow t1
  - s2 \rightarrow t2
  - s3 \rightarrow t3
  - s4 \rightarrow t4

**Conventional name:** partial function
Multiplicities can be applied to the domain, range or both of a relation.

- \( f : S \rightarrow \text{one } T \)
  - says that, for each element \( s \) of \( S \), \( f \) maps \( s \) to exactly one value in \( T \)

Potential instances:

- Conventional name: total function
Multiplicities can be applied to the domain, range or both of a relation.

- **$f : S \text{ one } \rightarrow \text{ one } T$**
  - says that, for each element $t$ of $T$, exactly one element of $S$ is mapped to $t$ by $f$ (plus earlier constraints)

Potential instances:

- **Conventional name:** bijection
Multiplicities and Relations

Multiplicities can be applied to the domain, range or both of a relation.

- **f : S → T**
  - says that, for each element \( t \) of \( T \), at most one element of \( S \) is mapped to \( t \) by \( f \) (plus earlier constraints)

- Potential instances:

  s1 \( \rightarrow \) t1 \( \checkmark \)  s1 \( \rightarrow \) t1 \( \checkmark \)  s1 \( \rightarrow \) t1 \( \times \)  s1 \( \rightarrow \) t1
  s2 \( \rightarrow \) t2 \( \times \)  s2 \( \rightarrow \) t2 \( \checkmark \)  s2 \( \rightarrow \) t2 \( \checkmark \)  s2 \( \rightarrow \) t2
  s3 \( \rightarrow \) t3 \( \times \)  s3 \( \rightarrow \) t3 \( \checkmark \)  s3 \( \rightarrow \) t3 \( \checkmark \)  s3 \( \rightarrow \) t3
  s4 \( \rightarrow \) t4 \( \times \)  s4 \( \rightarrow \) t4 \( \checkmark \)  s4 \( \rightarrow \) t4 \( \checkmark \)  s4 \( \rightarrow \) t4

*Conventional name:* partial injective function
Multiplicities can be applied to the domain, range or both of a relation.

- Other kinds of relational structures can be specified using multiplicities

Examples:
- \( f : S \text{ lone} \rightarrow T \) ...partial injective relation
- \( f : S \rightarrow \text{ some } T \) ...total relation
- \( f : S \text{ some} \rightarrow T \) ...surjective relation
Example: Family Structure

- How would you use multiplicities to define the *wife* relation?

  
  \[
  \text{wife} : \text{Man lone} \rightarrow \text{lone Woman}
  \]

  
  - Intuition: partial injective function
    - Each man has zero or one woman as a wife
    - Each wife has zero or one man as a husband
Facts: general constraints

- Multiplicities constrain a single relation
- Facts can constrain interaction among several

```plaintext
sig Person { mom : Person }

fact {
    husband = ~wife
}
```
module cs6610/Family

abstract sig Person {
    siblings : set Person,
    children : set Person,
    parents : set Person
}
sig Man extends Person {
    wife : lone Woman
}
sig Woman extends Person {
    husband : lone Man
}
sig Married in Person {}

fact { wife = ~husband }

fact { parents = ~children }
Demo: Alloy Constraint Analyzer

- Load, compile, analyze cycle
- Commands
  - pred P() {}
  - run P for 4
  - Find an instance model, including predicate P, with scope 4
- Visualization of model instances
- Increase/decrease scopes
For you to do

- Now you should startup Alloy
- Compile it
- Analyze the state schema
- Look at the generated instance
- Does it look correct?
- What if anything would you change about it?
Instance found:

Domains:
   Person = \{P0,P1,P2\}

Sets:
   Man = \{P1,P2\}
   Married = \{P0,P1,P2\}
   Woman = \{P0\}

Relations:
   children = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P0,P1,P2\}, P2 \rightarrow \{P2\}\}
   husband = \{P0 \rightarrow \{P2\}\}
   parents = \{P0 \rightarrow \{P0,P1\}, P1 \rightarrow \{P1\}, P2 \rightarrow \{P0,P1,P2\}\}
   siblings = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P2\}, P2 \rightarrow \{P0,P1,P2\}\}
   wife = \{P2 \rightarrow \{P0\}\}
Person can be their own parent?

Instance found:
Domains:
    Person = \{P0, P1, P2\}
Sets:
    Man = \{P1, P2\}
    Married = \{P0, P1, P2\}
    Woman = \{P0\}
Relations:
    children = \{P0 \rightarrow \{P0, P2\}, P1 \rightarrow \{P0, P1, P2\}, P2 \rightarrow \{P2\}\}
    husband = \{P0 \rightarrow \{P2\}\}
    parents = \{P0 \rightarrow \{P0, P1\}, P1 \rightarrow \{P1\}, P2 \rightarrow \{P0, P1, P2\}\}
    siblings = \{P0 \rightarrow \{P0, P2\}, P1 \rightarrow \{P2\}, P2 \rightarrow \{P0, P1, P2\}\}
    wife = \{P2 \rightarrow \{P0\}\}
Instance found:
Domains:
  Person = \{P0, P1, P2\}
Sets:
  Man = \{P1, P2\}
  Married = \{P0, P1, P2\}
  Woman = \{P0\}
Relations:
  children = \{P0 \rightarrow \{P0, P2\}, P1 \rightarrow \{P0, P1, P2\}, P2 \rightarrow \{P2\}\}
  husband = \{P0 \rightarrow \{P2\}\}
  parents = \{P0 \rightarrow \{P0, P1\}, P1 \rightarrow \{P1\}, P2 \rightarrow \{P0, P1, P2\}\}
  siblings = \{P0 \rightarrow \{P0, P2\}, P1 \rightarrow \{P2\}, P2 \rightarrow \{P0, P1, P2\}\}
  wife = \{P2 \rightarrow \{P0\}\}
Self-Siblings, Child-Siblings?

Instance found:
Domains:
    Person = \{P0,P1,P2\}
Sets:
    Man = \{P1,P2\}
    Married = \{P0,P1,P2\}
    Woman = \{P0\}
Relations:
    children = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P0,P1,P2\}, P2 \rightarrow \{P2\}\}
    husband = \{P0 \rightarrow \{P2\}\}
    parents = \{P0 \rightarrow \{P0,P1\}, P1 \rightarrow \{P1\}, P2 \rightarrow \{P0,P1,P2\}\}
    siblings = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P2\}, P2 \rightarrow \{P0,P1,P2\}\}
    wife = \{P2 \rightarrow \{P0\}\}
Instance found:
Domains:
   Person = \{P0,P1,P2\}
Sets:
   Man = \{P1,P2\}
   Married = \{P0,P1,P2\}
   Woman = \{P0\}
Relations:
   children = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P0,P1,P2\}, P2 \rightarrow \{P2\}\}
   husband = \{P0 \rightarrow \{P2\}\}
   parents = \{P0 \rightarrow \{P0,P1\}, P1 \rightarrow \{P1\}, P2 \rightarrow \{P0,P1,P2\}\}
   siblings = \{P0 \rightarrow \{P0,P2\}, P1 \rightarrow \{P2\}, P2 \rightarrow \{P0,P1,P2\}\}
   wife = \{P2 \rightarrow \{P0\}\}
Model Weaknesses

- The model is underconstrained
  - It doesn’t match our domain knowledge
  - We can add constraints to enrich the model
- Underconstrained models are common early in the development process
  - Alloy gives us quick feedback on weaknesses in our model
  - We can incrementally add constraints until we are satisfied with it
Adding Constraints

- We’d like to enforce the following constraints which are simply matters of biology
  - No person can be their own parent (or more generally, their own ancestor)
  - No person can have more than one father or mother
  - A person’s siblings are those with the same parents
Adding Constraints

- We’d like to enforce the following social constraints
  - Any married man has a wife
  - Any married woman has a husband
  - A man’s wife cannot be one of his siblings