Métodos Formais
2023.2

Alloy modeling: Academia model

Área de Teoria DCC/UFMG

“Academia” modeling example

- We will model an academic enterprise expressing relationships between
- People
  - Faculty
  - Students
    - Graduate
    - Undergraduate
  - Instructors
- Courses

How should we model these basic domains in Alloy?

Strategy

- Build and validate your model incrementally
  - Start with basic signatures and fields
  - Add basic constraints
  - Instantiate the model and study the results
  - Probe the model with assertions

- Add groups of features at a time
  - New signatures and fields
  - New constraints
  - Confirm previous assertions
  - Probe new features with assertions

Basic Components

- People
  - Students: Undergrads and Grads
  - Instructors: Faculty and Grads
- Courses
- Relationships
  - One instructor teaches a course
  - One or more students are taking a course
  - Students can be waiting for a course

Auxiliary relations

- We may choose to define auxiliary relations:
  - `teaches` (transpose of `taughtby`)
  - `taking` (transpose of `enrolled`)
  - `waitingfor` (transpose of `waitlist`)

```
fun teaches: Instructor -> Course { ~taughtby }
fun taking: Student -> Course { ~enrolled }
fun waitingfor: Student -> Course { ~waitlist }
```

- Or not:
  - if `i` is an instructor, then

```
i.teaches <=> taughtby.i
```

Academia constraints

- All instructors are either faculty or graduate students
  - Was not expressed in set definition–although it could have, with

```
sig Instructor in Graduate + Faculty
```

- No one is waiting for a course unless someone is enrolled

- No graduate students teach a course that they are enrolled in or waiting for

Academia constraints

- All instructors are either faculty or graduate students
  - Was not expressed in set definition–although it could have, with

```
sig Instructor in Graduate + Faculty
```

  As a fact:

```
all i: Instructor | i in Faculty + Graduate
```

- No one is waiting for a course unless someone is enrolled

- No graduate students teach a course that they are enrolled in or waiting for

Academia constraints

- All instructors are either faculty or graduate students
  - Was not expressed in set definition–although it could have, with

```
sig Instructor in Graduate + Faculty
```

  As a fact:

```
all i: Instructor | i in Faculty + Graduate
```

- No one is waiting for a course unless someone is enrolled

```
all c: Course |
    some c.waitlist => some c.enrolled
```

  Actually superfluous. Why?

- No graduate students teach a course that they are enrolled in or waiting for

Academia constraints

- All instructors are either faculty or graduate students
  - Was not expressed in set definition–although it could have, with

```
sig Instructor in Graduate + Faculty
```

  As a fact:

```
all i: Instructor | i in Faculty + Graduate
```

- No one is waiting for a course unless someone is enrolled

```
all c: Course |
    some c.waitlist => some c.enrolled
```

  Actually superfluous. Why?

- No graduate students teach a course that they are enrolled in or waiting for

```
all c: Course |
    c.taughtby !in c.enrolled + c.waitlist
```

Academia *realism* constraints

To make instances more interesting to analyze, we can add "realism" facts or constraints in the `run` command:

- There is a graduate student who is an instructor

- There are at least:
  - Two courses and
  - Three undergraduates

Academia *realism* constraints

To make instances more interesting to analyze, we can add "realism" facts or constraints in the `run` command:

- There is a graduate student who is an instructor
- There are at least:
  - Two courses and
  - Three undergraduates
- We can also define a predicate:

```
pred RealismConstraints [] {
  some Graduate & Instructor
  #Course > 1
  #Undergrad > 2
}
```

Academia *realism* constraints

- No instances exist in the default scope is an instructor

- Why?
  - default scope is up to 3 elements in top-level sigs
  - So we cannot have more than 3 students

- The constraints

```
some Graduate & Instructor
#Undergrad > 2
```

  entail at least 4 students

Academia assertions

- No student is enrolled and on the waitlist for the same course

- No instructor is on the waitlist for a course that they teach

Academia assertions

- No student is enrolled and on the waitlist for the same course

```
assert NoEnrolledAndWaiting {
  all c: Course |
        no (c.enrolled & c.waitlist)
}
```

- No instructor is on the waitlist for a course that they teach

Academia assertions

- No student is enrolled and on the waitlist for the same course

```
assert NoEnrolledAndWaiting {
  all c: Course |
        no (c.enrolled & c.waitlist)
}
```

- No instructor is on the waitlist for a course that they teach

```
assert NoWaitingTeacher {
  all c: Course |
        no (c.taughtby & c.waitlist)
}
```

Academia assertions

- No student is enrolled and on the waitlist for the same course
  - A counterexample has been found, hence *we transform this assertion into a fact*.

- No instructor is on the waitlist for a course that they teach
  - No counterexamples. So is it valid?

Academia assertions

- No student is enrolled and on the waitlist for the same course
  - A counterexample has been found, hence *we transform this assertion into a fact*.

- No instructor is on the waitlist for a course that they teach
  - No counterexamples. So is it valid?
  - Not necessarily! But we can generally rely on the *small scope hypothesis*:
    - if an assertion is not valid, it probably has a small counter-example
  - But why is this assertion valid?

Academia assertions

- No student is enrolled and on the waitlist for the same course
  - A counterexample has been found, hence *we transform this assertion into a fact*.

- No instructor is on the waitlist for a course that they teach
  - No counterexamples. So is it valid?
  - Not necessarily! But we can generally rely on the *small scope hypothesis*:
    - if an assertion is not valid, it probably has a small counter-example
  - But why is this assertion valid?
    - Since faculty are not students, they cannot be on a waitlist

Academia assertions

- No student is enrolled and on the waitlist for the same course
  - A counterexample has been found, hence *we transform this assertion into a fact*.

- No instructor is on the waitlist for a course that they teach
  - No counterexamples. So is it valid?
  - Not necessarily! But we can generally rely on the *small scope hypothesis*:
    - if an assertion is not valid, it probably has a small counter-example
  - But why is this assertion valid?
    - Since faculty are not students, they cannot be on a waitlist
    - Grad students do not teach courses they are enrolled in or waiting to enroll in

Extensions

- Add an attribute for students
  - Unique IDs
  - Note you'll need a new signature
- Add student transcripts (only taken courses, no grades)
  - A student's transcript contains a course only if it contains the course's prerequisites
- Add prerequisite structure for courses
  - A courses does not have itself as a prerequisite
  - Students can only wait to be in a course for which they already have the prerequisites
- Do a realism predicate where there exists a course with prerequisites and with students enrolled.

Acknowledgments

These notes are heavily based on notes from Matt Dwyer, John Hatcliff, Rod Howell, Laurence Pilard and Cesare Tinelli.