In the type system, multiplicities are modeled orthogonal to each other. This works out well because these are orthogonal issues.

Multiplicities on relations and attributes remove the need for collections and nullable types.

There are four multiplicities:

- \([0,1]\): symbol required
- \([1]\): symbol required
- \([0,\infty]\): symbol + one or more
- \([1,\infty]\): symbol + one or more

Declarative specification of derived values removes code for control flow and caching.

There are three attribute types:

- **Nominal**: no derivation, values can always be assigned
- **Default**: if a value is assigned, then this is returned, else the computed value is returned
- **Derivation**: no value can be assigned, the computed value is returned

Prototype

```
entity Student {
  name : String
  avgGrade : Int = avg(
    this * enrollment . grade
  )
}

entity Course {
  name : String
}

relation Enrollment {
  Student + Course +
  grade : Int?
  late : Int = \(0\) (default value)
  pass : Boolean =
  this . grade - this . late >= 6
  =>
}

relation Mentoring {
  Student + Enrollment +
}
```

Shorthand Relation Notation

The navigation names can be automatically derived if there are no name collisions. A programmer can also manually define names, and has to do so in the case of name collisions.

```
relation Enrollment {
  Student * Course +
}
```

Expands to:

```
relation Enrollment {
  Student . student + enrollment
  Course . course + enrollment
  course . student . student . course
}
```

Relation Navigation

There are three sorts of names defined to navigate:

- **Roles**: names in relation referring to participants
- **Inverses**: names in participant to relations
- **Shortcuts**: names in participants referring to other participants in relation

Future Work

- **Type-and-Multiplicity-safe operations**: edit data type-safe and preserving multiplicity constraints
- **Generalise multiplicities**: currently operations with multiplicities are built in, allow users to define these
- **Extend type system orthogonally**: next to type and multiplicity add ordering, allow duplicates, etcetera