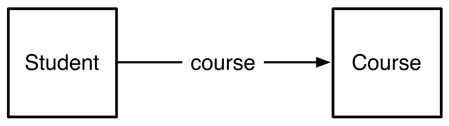
Relational Programming Languages

Daco C. Harkes, Eelco Visser
Delft University of Technology, The Netherlands
{d.c.harkes,e.visser}@tudelft.nl

Design Space for Relations

Object Oriented
Edge name defined in source namespace

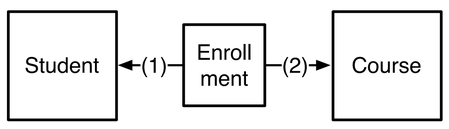
Edge — Object-Oriented Reference




```
class Student {
  Course* course;
}
class Course { }

bob.course => math
```

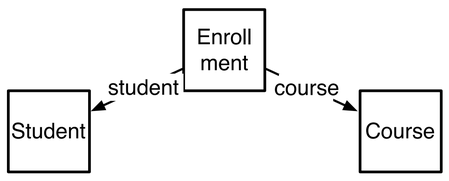
Tuple — Object-Oriented Tuple




```
class Student { }
class Course { }
class Enrollment extends
  Pair<Student, Course> { }

b_takes_m.first  => bob
b_takes_m.second => math
```

Object — Object-Oriented Class



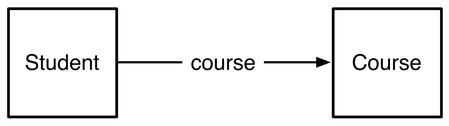

```
class Student { } class Course { }
class Enrollment {
  Student student
  Course  course
}

b_takes_m.student => bob
b_takes_m.course  => math
```

Concise Navigation

Relational (SQL), ER, UML
Edge name defined in source + Inverse lookup

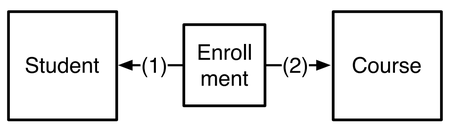
Edge — Inverse Reference Navigation




```
entity Student {
  Course* course+
}
entity Course { }

bob.course             => math
math<-(Student.course) => bob
```

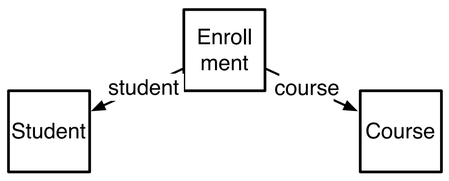
Tuple — Relations as Tuples [RelJ]




```
entity Student { } entity Course { }
relation Enrollment <*Student, +Course>

bob.Enrollment   => math
bob:Enrollment   => b_takes_m
b_takes_m.from   => bob
b_takes_m.to     => math
```

Object — Relation Objects [Rumer, RelJ e]



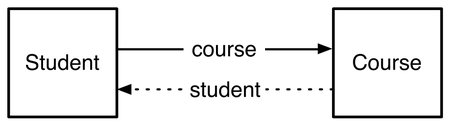

```
entity Student { } entity Course { }
relation Enrollment {
  Student student *
  Course  course  +
}

bob<-(Enrollment.student).course  => math
bob<-(Enrollment.student)         => b_takes_m
math<-(Enrollment.course).student => bob
math<-(Enrollment.course)         => b_takes_m
b_takes_m.student                 => bob
b_takes_m.course                  => math
```

Concise Navigation

Object Role Modeling
Edge defined in source and target w. diff. names

Edge — Inverse Properties [WebDSL]



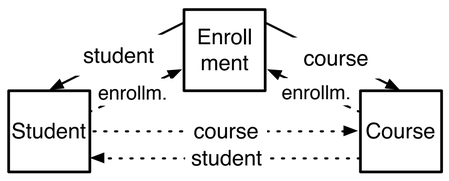

```
entity Student {
  Course* course <- + student
}
entity Course { }

bob.course    => math
math.student  => bob
```

Tuple

It does not make sense to define inverse reference names without role names

Object — Relations w. Concise Navigation

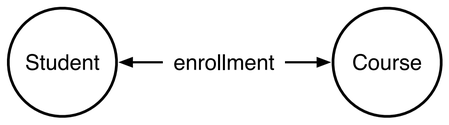



```
entity Student { } entity Course { }
relation Enrollment {
  Student student <- * enrollment
  Course  course  <- + enrollment
  student.course  <->  course.student
}
bob.course         => math
bob.enrollment     => b_takes_m
math.student       => bob
math.enrollment    => b_takes_m
b_takes_m.student  => bob
b_takes_m.course   => math
```

Concise Navigation

Graph Databases
Edge defined in source and target w. same name

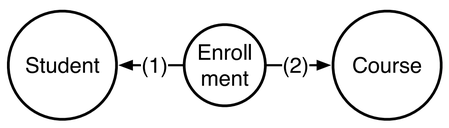
Edge — Undirected Graph




```
entity Student {
  Course* enrollment +
}
entity Course  { }

bob.enrollment  => math
math.enrollment => bob
```

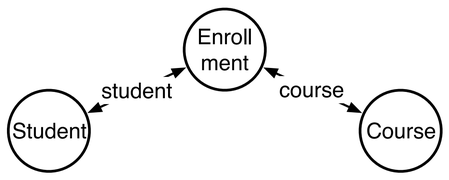
Tuple — Intermediary Nodes




```
entity Student { } entity Course { }
relation Enrollment <*Student, +Course>

bob.Enrollment   => math
bob:Enrollment   => b_takes_m
b_takes_m.from   => bob
b_takes_m.to     => math
```

Object — Undirected indirect graph




```
entity Student { } entity Course { }
relation Enrollment {
  Student student *
  Course  course  +
}

bob.student.course  => math
bob.student         => b_takes_m
math.course.student => bob
math.course         => b_takes_m
b_takes_m.student   => bob
b_takes_m.course    => math
```

First-class citizenship

N-ary

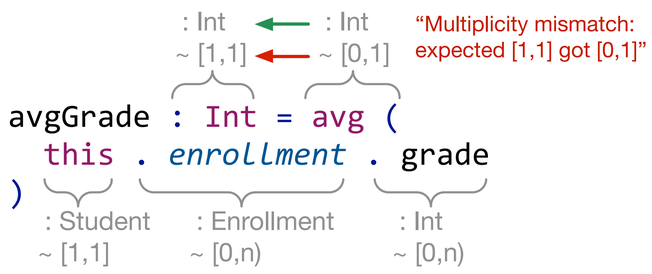
Concise Navigation

Bidirectional Navigation

Language Prototype

Type System

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



Multiplicities

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

There are four multiplicities:

- [0,1] symbol: **?** optional, nullable
- [1,1] symbol: required
- [0,n) symbol: ***** zero, one or more
- [1,n) symbol: **+** one or more

Derivations

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

There are three attribute types:

- *Normal*: no derivation, values can always be assigned
- *Default value*: 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
  <+
    false
}

relation Mentoring {
  Student mentor *
  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

[Rumer] Balzer, S.: *Rumer: a Programming Language and Modular Verification Technique Based on Relationships.* Ph.D. thesis (2011)
[RelJ] Bierman, G., Wren, A.: *First-class relationships in an object-oriented language.* ECOOP (2005)
Harkes, D. C., Visser, E.: *Unifying and Generalizing Relations in Role-Based Data Modeling and Navigation.* SLE (2014)
[WebDSL] Visser, E.: *WebDSL: A case study in domain-specific language engineering.* GTTSE (2007)
[RelJ e] Wren, A.: *Relationships for object-oriented programming languages.* Technical Report (2007)

TU Delft Delft University of Technology