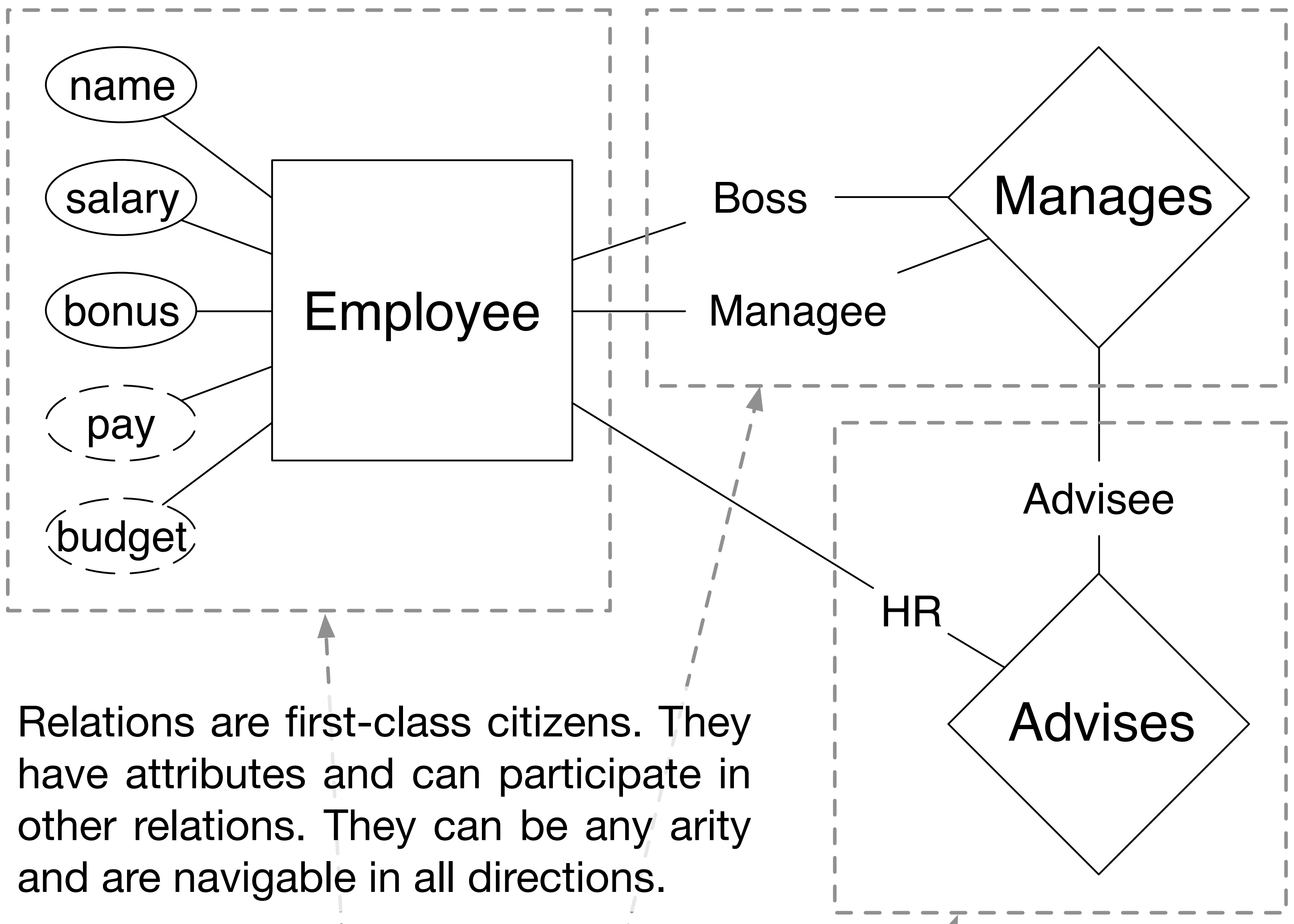
A Relational Programming Language

Daco Harkes
Delft University of Technology, The Netherlands
d.c.harkes@student.tudelft.nl

First-class citizen relations

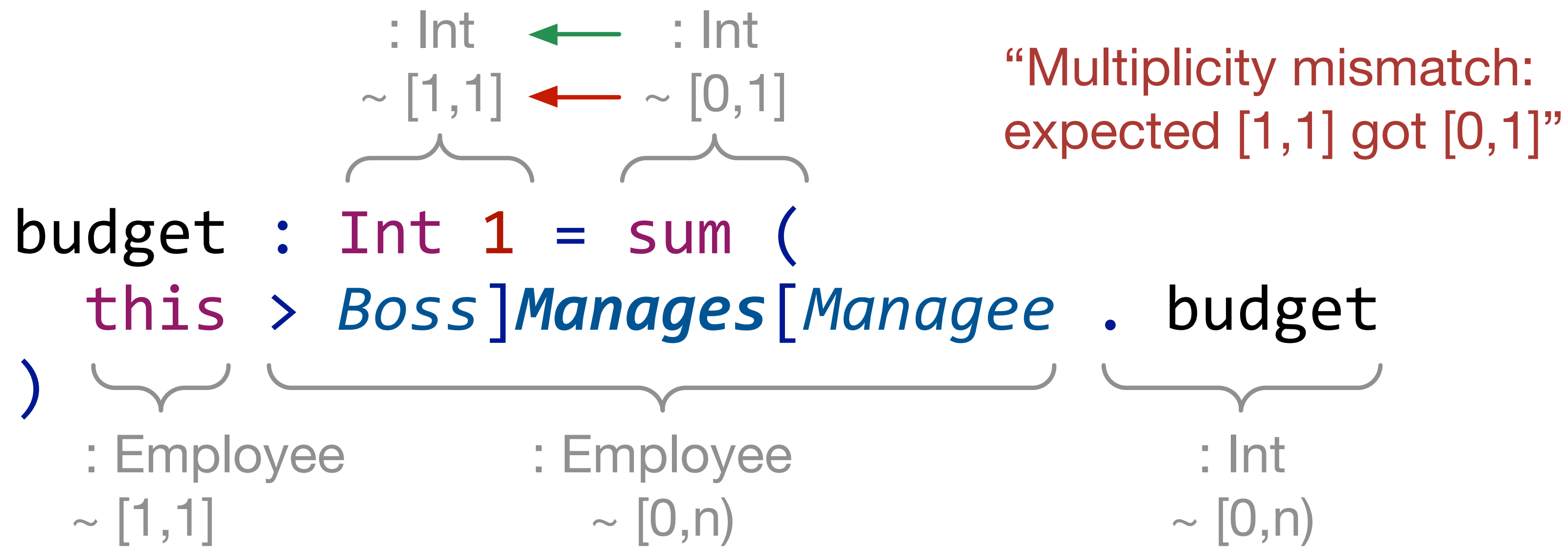


Relations are first-class citizens. They have attributes and can participate in other relations. They can be any arity and are navigable in all directions.

```
entity Employee {
  name   : String 1
  salary : Int 1
  bonus  : Int 1 = 0 (default value)
  pay    : Int 1 = salary + bonus

  budget : Int 1 =
    pay +
    (
      sum (
        this > Boss]Manages[Managee . budget
      )
      <+ 0
    )

}

relation Manages {
  Employee * Boss
  Employee ? Managee
}

relation Advises {
  Employee * HR
  Manages   1 Advisee
}
```

Type System

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



“Multiplicity mismatch: expected [1,1] got [0,1]”

```
budget : Int 1 = sum (
  this > Boss]Manages[Managee . budget
)
```

Problem

Problems in Object-Oriented Languages

- Pointers provide only one-way navigability
- Ternary relations requires lifting to objects
- Multiplicities require wrapping types in Collection containers

Problems in Relational Databases

- Hierarchies can only be saved and queried in normalized form
- Derived values, in views, do not (fully) support recursion
- One cannot build a program with just a Relational Database

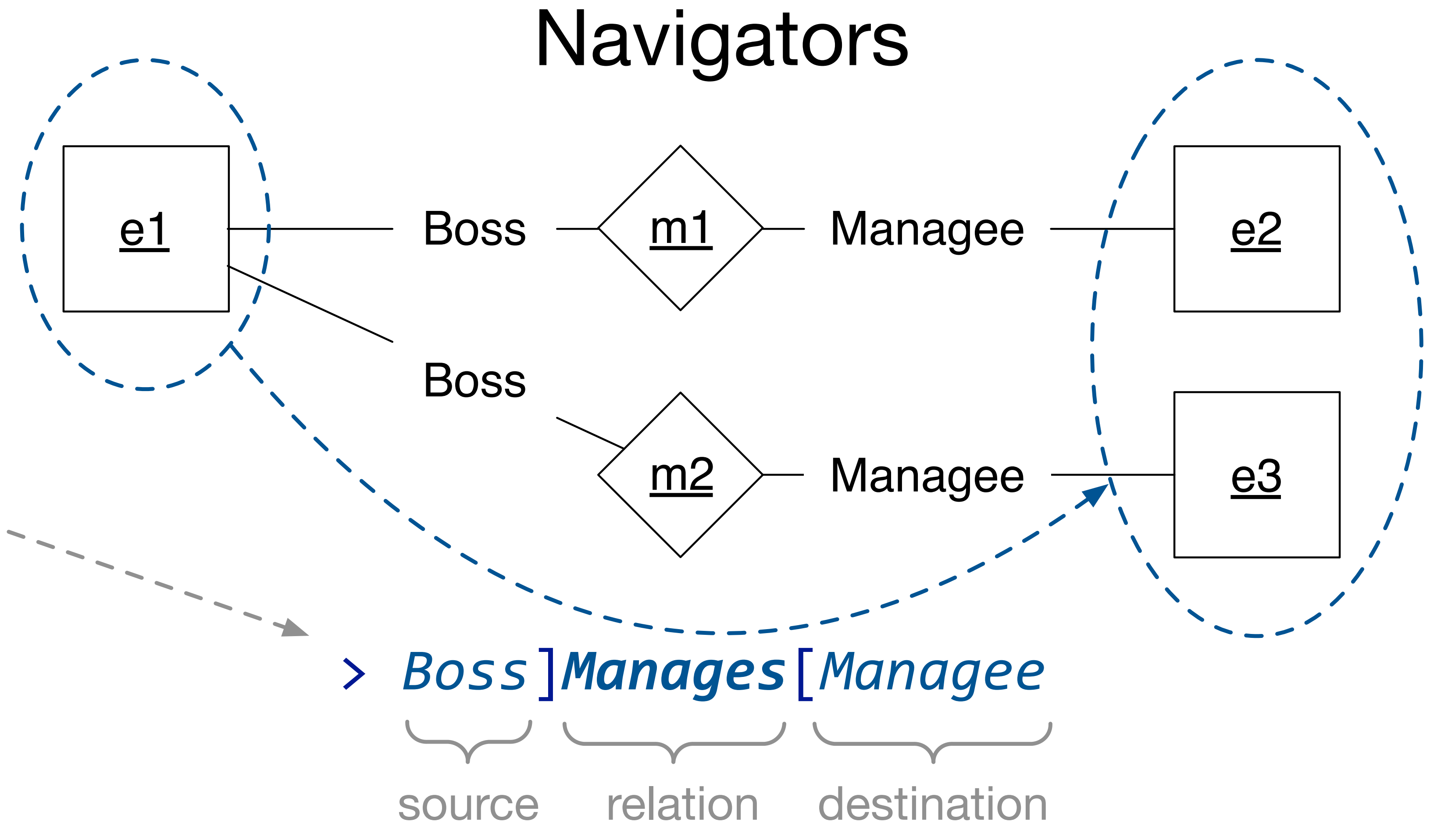
Derivations

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

Navigators



Multiplicities

Employee participates in Manages as Boss [0,n) times
Employee participates in Manages as Managee [0,1] times

Employee participates in Advises as HR [0,n) times
Manages participates in Advises as Advisee [1,1] times

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: 1	required
[0,n)	symbol: *	zero, one or more
[1,n)	symbol: +	one or more