Formal Specification of Software

The Object Contraint Language by Example

Bernhard Beckert

UNIVERSITÄT KOBLENZ-LANDAU

The Classifier Context

context **(c :)? typeName**

inv **expressionName? : OclExpression**

context **(c :)? typeName**

inv **expressionName$_1$? : OclExpression$_1$**

...

...

inv **expressionName$_n$? : OclExpression$_n$**

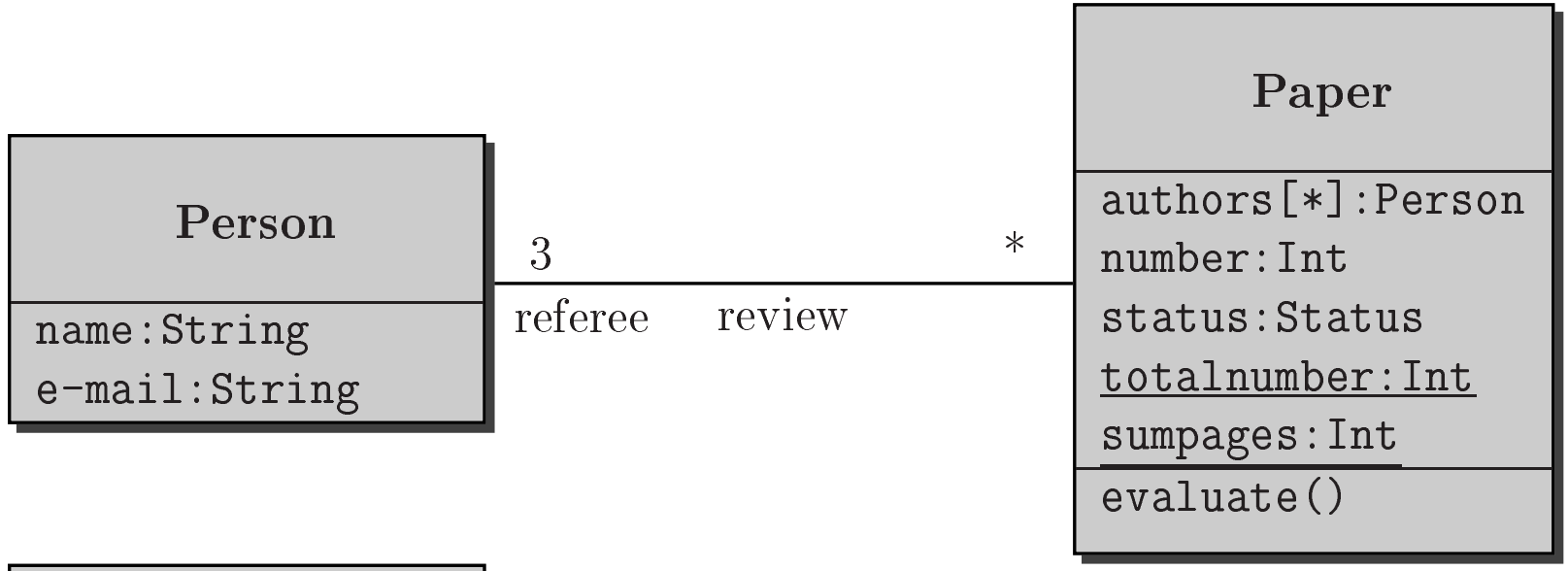
The Operator Context

context (c :)?

typeName ::opName(p_1: type$_1$; . . . ;p_k: type$_k$):rtype

{pre ,post } expressionName? : OclExpression

Constraints with Attributes



context **Paper**

inv **number** ≥ 1

Equivalent notational variations

context **Paper**

inv **self.number** ≥ 1

Equivalent notational variations

context Paper

inv self.number ≥ 1

context c:Paper

inv c.number ≥ 1

Equivalent notational variations

context Paper

inv self.number ≥ 1

context c:Paper

inv c.number ≥ 1

context c:Paper

inv startCount : c.number ≥ 1

Equivalent notational variations

context Paper

inv self.number ≥ 1

context c:Paper

inv c.number ≥ 1

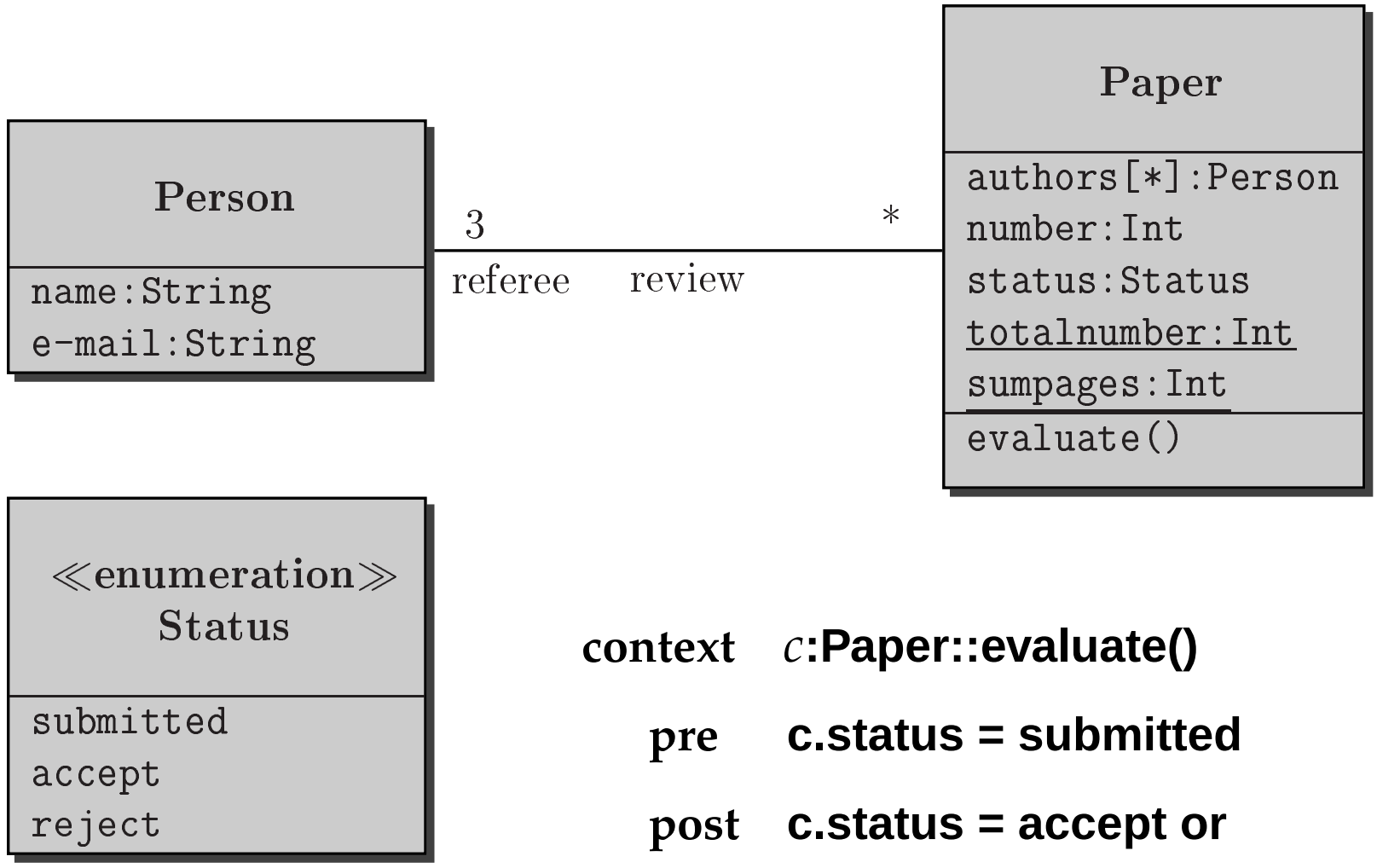
context c:Paper

inv startCount : c.number ≥ 1

context Paper

inv startCount : number ≥ 1

Operator Constraint



context c**:Paper::evaluate()**

pre **c.status = submitted**

post **c.status = accept or**

c.status = reject

Types

Model types

The classes form the context diagram of an OCL constraint

Types

Model types

The classes form the context diagram of an OCL constraint

Basic types

Integer, *Real*, *Boolean* **and** *String*

Types

Model types

The classes form the context diagram of an OCL constraint

Basic types

Integer, *Real*, *Boolean* **and** *String*

Enumeration types

The user defined enumeration types

Types

Model types

The classes form the context diagram of an OCL constraint

Basic types

Integer, *Real*, *Boolean* **and** *String*

Enumeration types

The user defined enumeration types

Collection types

Set, *Bag*, *Sequence*

Types

Model types

The classes form the context diagram of an OCL constraint

Basic types

Integer, *Real*, *Boolean* **and** *String*

Enumeration types

The user defined enumeration types

Collection types

Set, *Bag*, *Sequence*

Special types

e.g. *OclAny*, *OclType*

Subtyping

- T_1, T_2 **model types:**

  $T_1 < T_2$ **holds exactly if** T_1 **is a subclass of** T_2

Subtyping

- T_1, T_2 **model types:**

  $T_1 < T_2$ **holds exactly if** T_1 **is a subclass of** T_2

- $Integer < Real$

Subtyping

- T_1, T_2 **model types:**

  $T_1 < T_2$ **holds exactly if** T_1 **is a subclass of** T_2

- $Integer < Real$

- **For all type expressions** T**, not denoting a collection type:**
  - $Set(T) < Collection(T)$
  - $Bag(T) < Collection(T)$
  - $Sequence(T) < Collection(T)$

Subtyping

- T_1, T_2 **model types:**

  $T_1 < T_2$ **holds exactly if** T_1 **is a subclass of** T_2

- $Integer < Real$

- **For all type expressions** T**, not denoting a collection type:**
  - $Set(T) < Collection(T)$
  - $Bag(T) < Collection(T)$
  - $Sequence(T) < Collection(T)$

- **If** T **is a model, basic, or enumeration type:** $T < OCLAny$

Subtyping

- T_1, T_2 **model types:**

  $T_1 < T_2$ **holds exactly if** T_1 **is a subclass of** T_2

- $Integer < Real$

- **For all type expressions** T**, not denoting a collection type:**

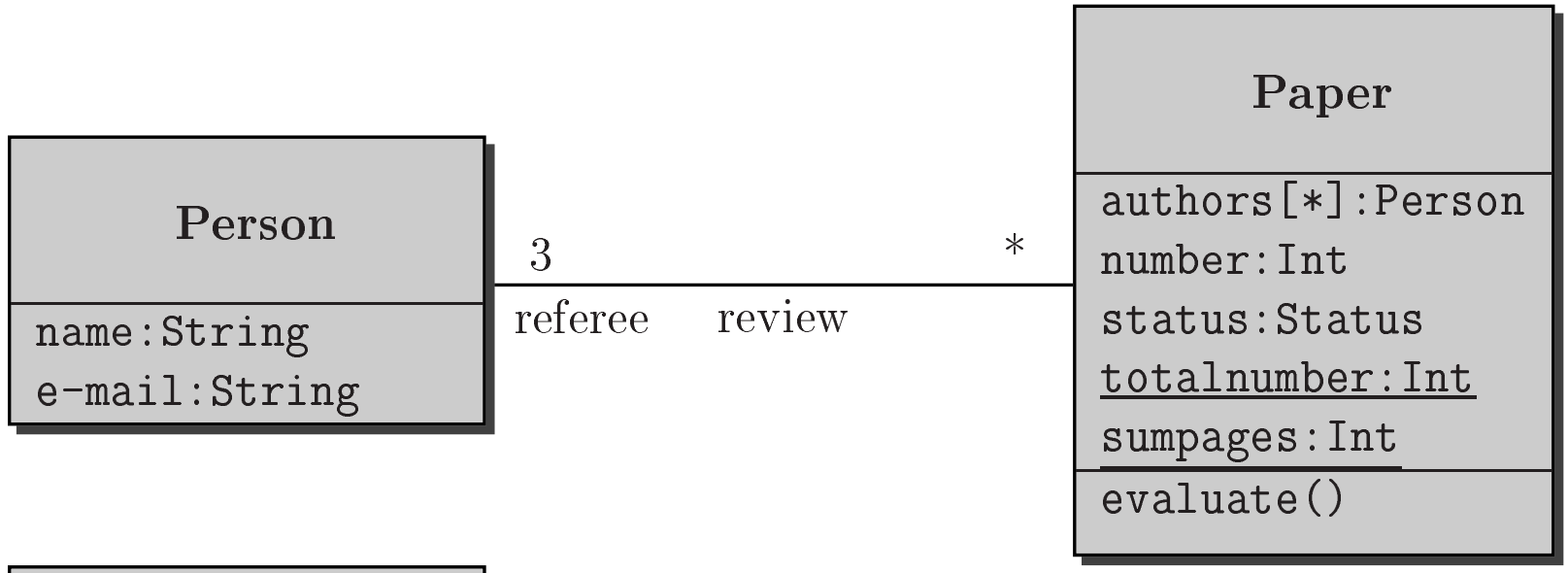
  - $Set(T) < Collection(T)$
  - $Bag(T) < Collection(T)$
  - $Sequence(T) < Collection(T)$

- **If** T **is a model, basic, or enumeration type:** $T < OCLAny$

- **If** $T_1 < T_2$ **and** C **is any of the type constructors** $Collection$**,** Set**,** Bag**,** $Sequence$**:**

  $C(T_1) < C(T_2)$.

Typing Examples



constraint p:Person

$p.name$, $p.$**e-mail have type** $String$**.**

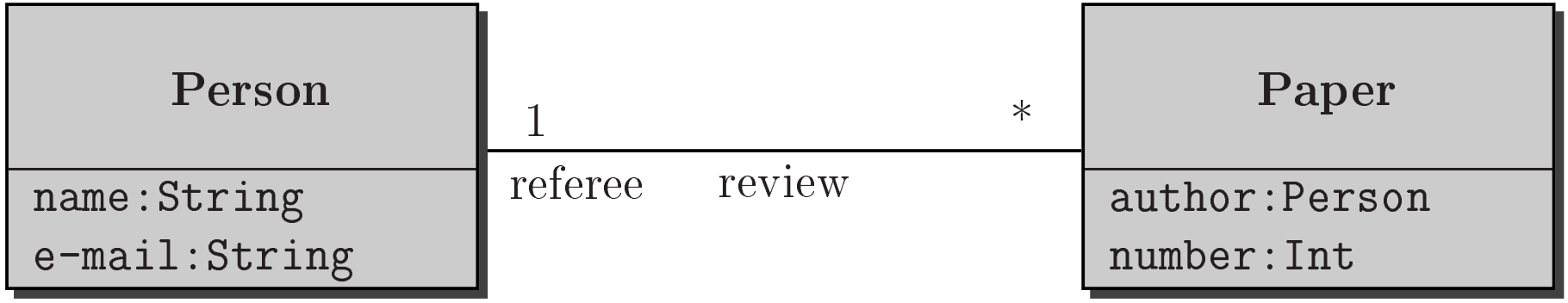
constraint c:Paper

$c.number$ **has type** $Integer$**,**

$c.status$ **has type** $Status$**,**

$c.authors$ **has type** $Set(Person)$

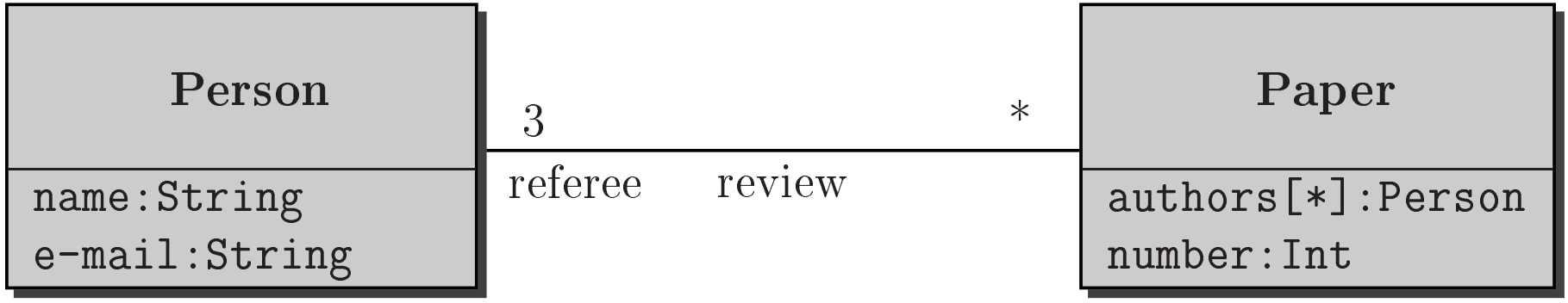
Constraints with Associations



context **c:Paper**

inv **c.author <> c.referee**

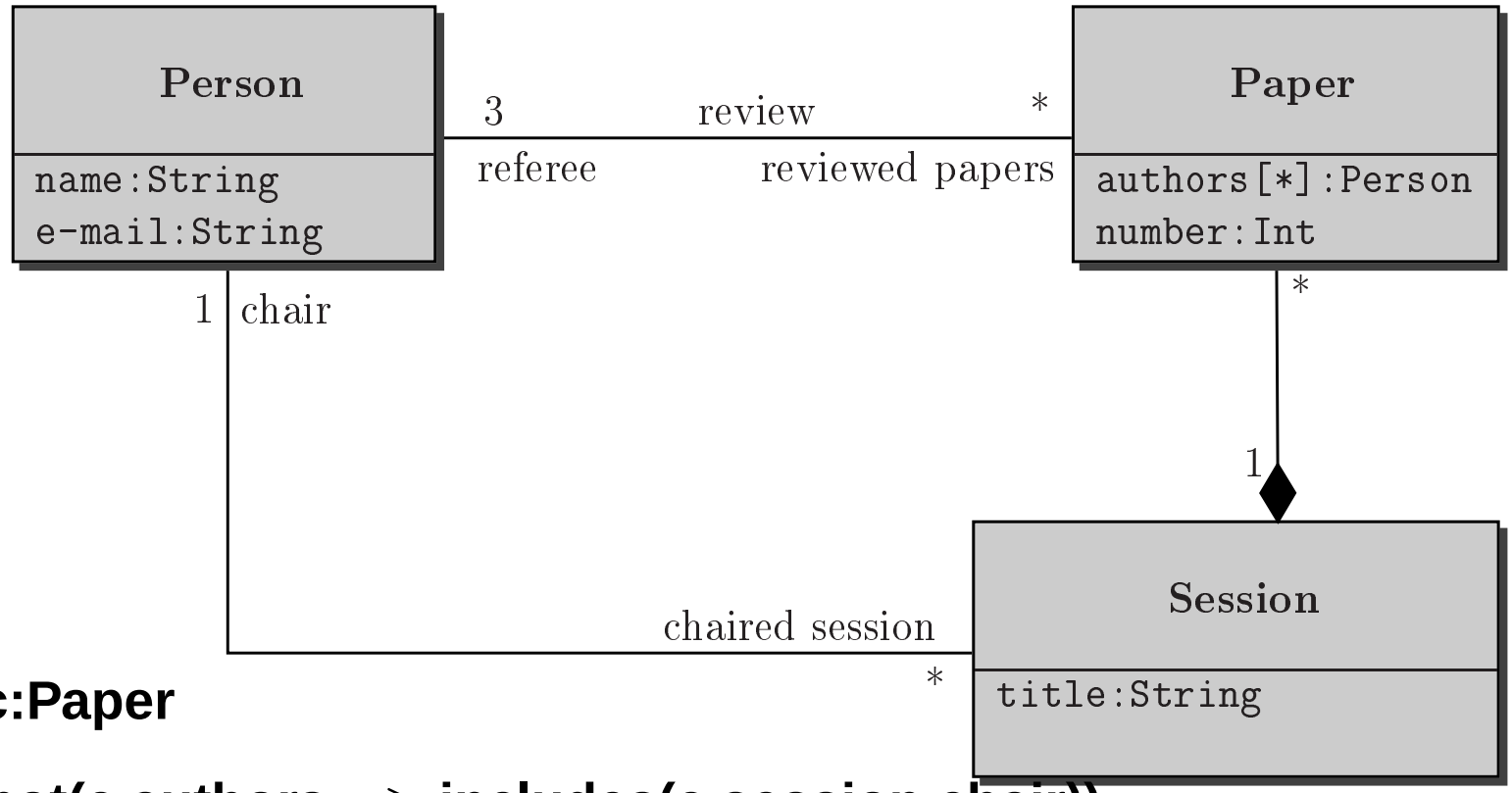
Constraints with Associations



context **c:Paper**

inv **c.authors –> intersection(c.referee) –> isEmpty**

Constraints and Navigation



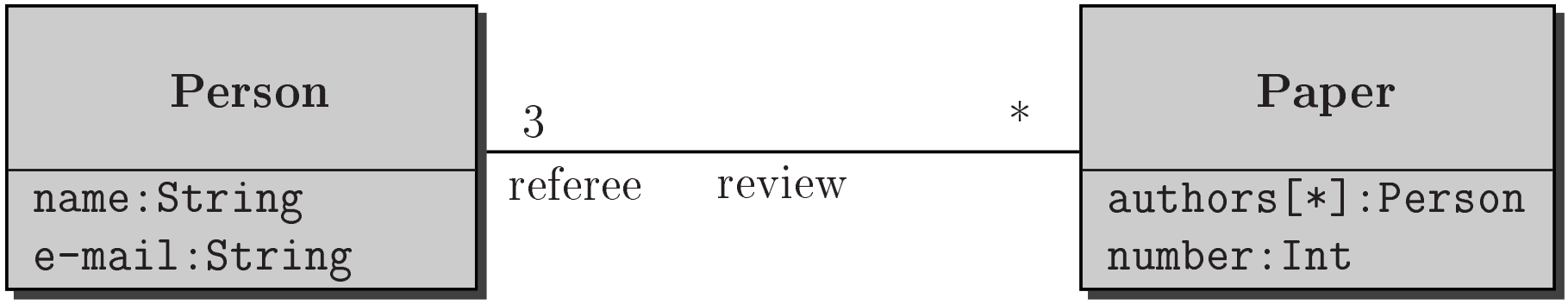
context c:Paper

inv not(c.authors –> includes(c.session.chair))

context p:Person

inv p.reviewed_papers.session.chair –> includes(p)

allInstances



context **Person**

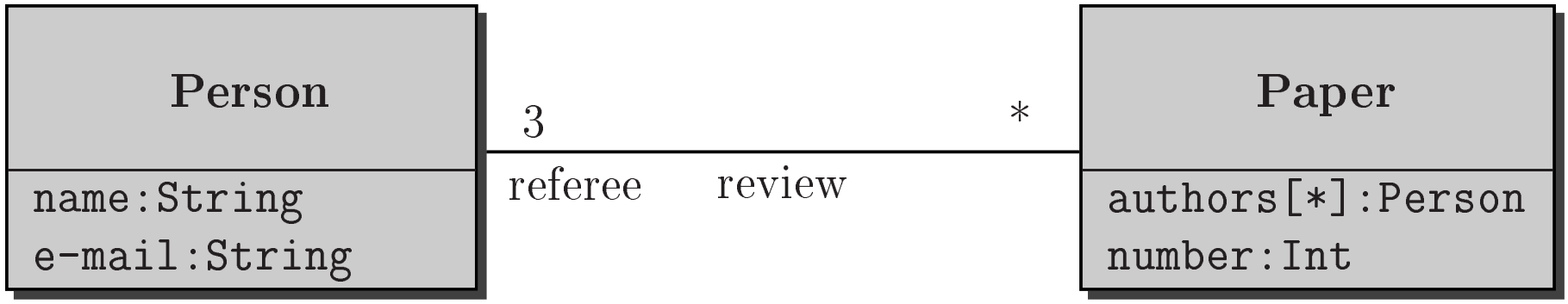
inv **Person.allInstances –> forAll(p | p.e-mail.size $\geq$ 3)**

context **Paper**

inv **Paper.allInstances –> forAll(p1, p2 |**

p1 <> p2 implies p1.number <> p2.number)

Avoiding *allInstances*



context **Person**

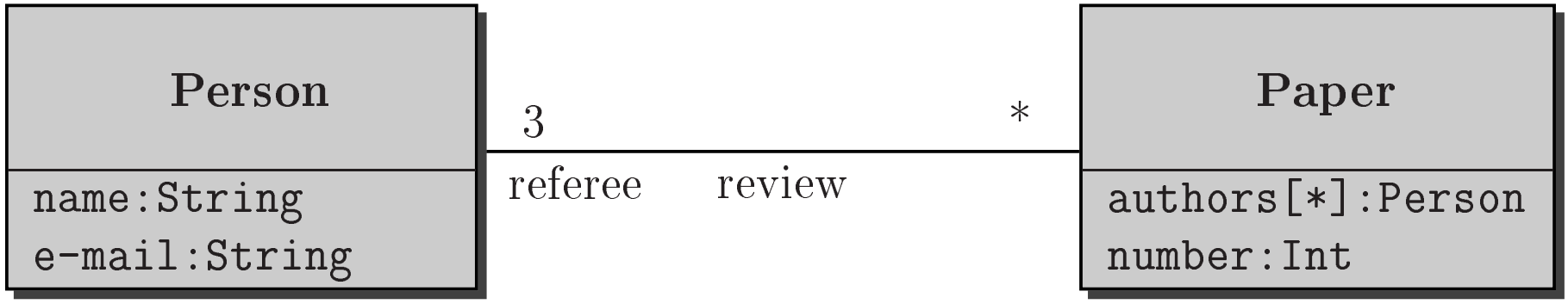
inv **Person.allInstances –> forAll(p | p.e-mail.size $\geq$ 3)**

Can be equivalently replaced by:

context **p:Person**

inv **p.e-mail.size $\geq$ 3**

Avoiding *allInstances*



context **Paper**

inv **Paper.allInstances –> forAll(p1, p2 |**

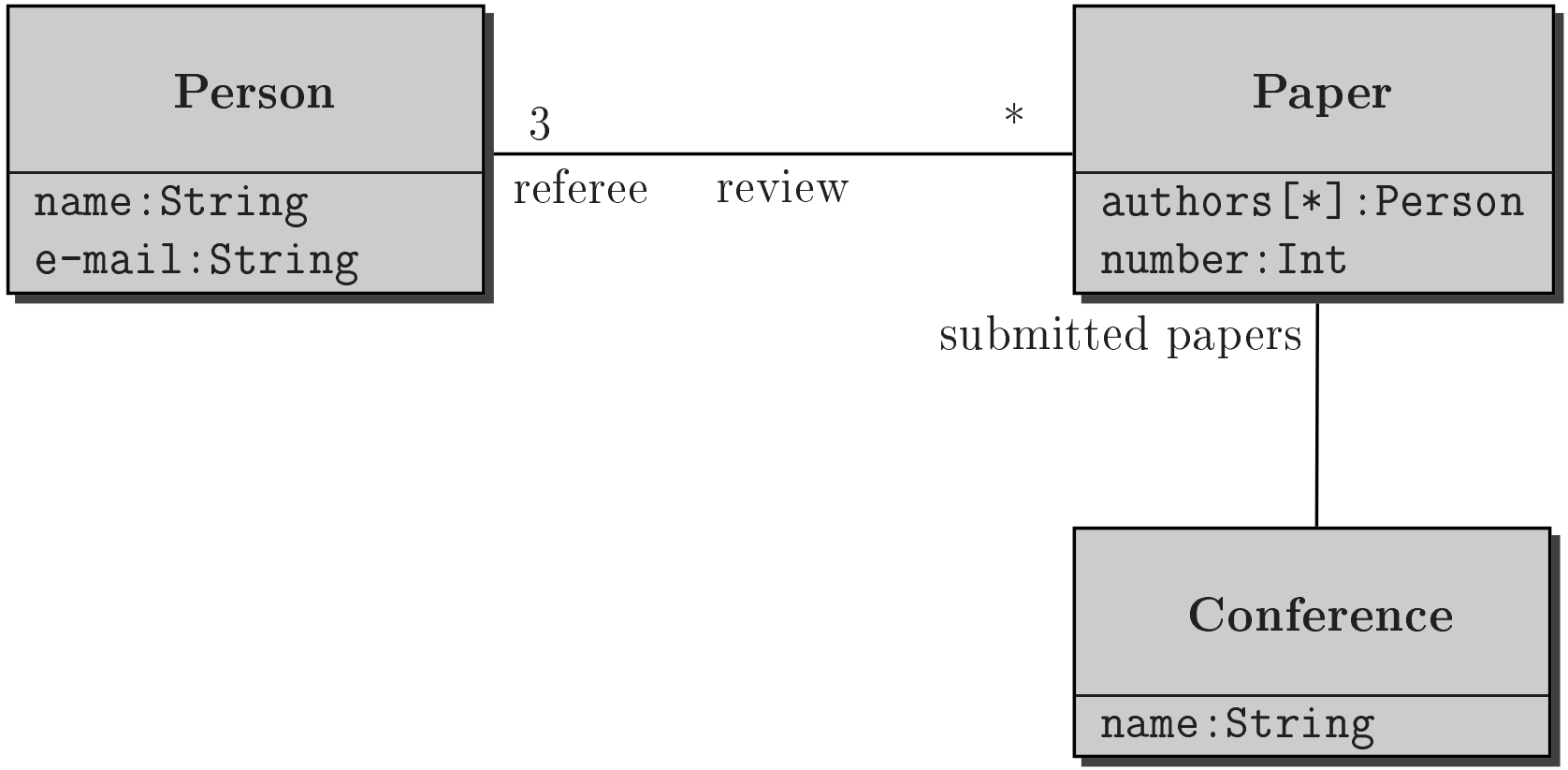
p1 <> p2 implies p1.number <> p2.number)

Can be equivalently replaced by:

context **p1,p2:Papers**

inv **p1 <> p2 implies p1.number <> p2.number)**

Avoiding *allInstances*

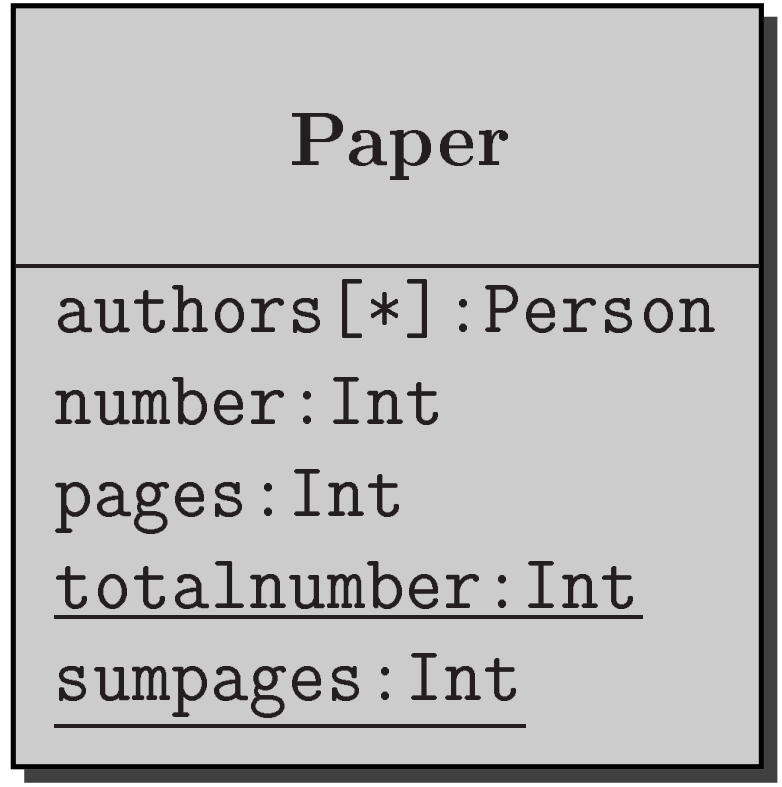


context **Conference**

inv **self.submitted_papers –> forAll(p1, p2 |**

p1 <> p2 implies p1.number <> p2.number)

Introducing the *iterate* Operation



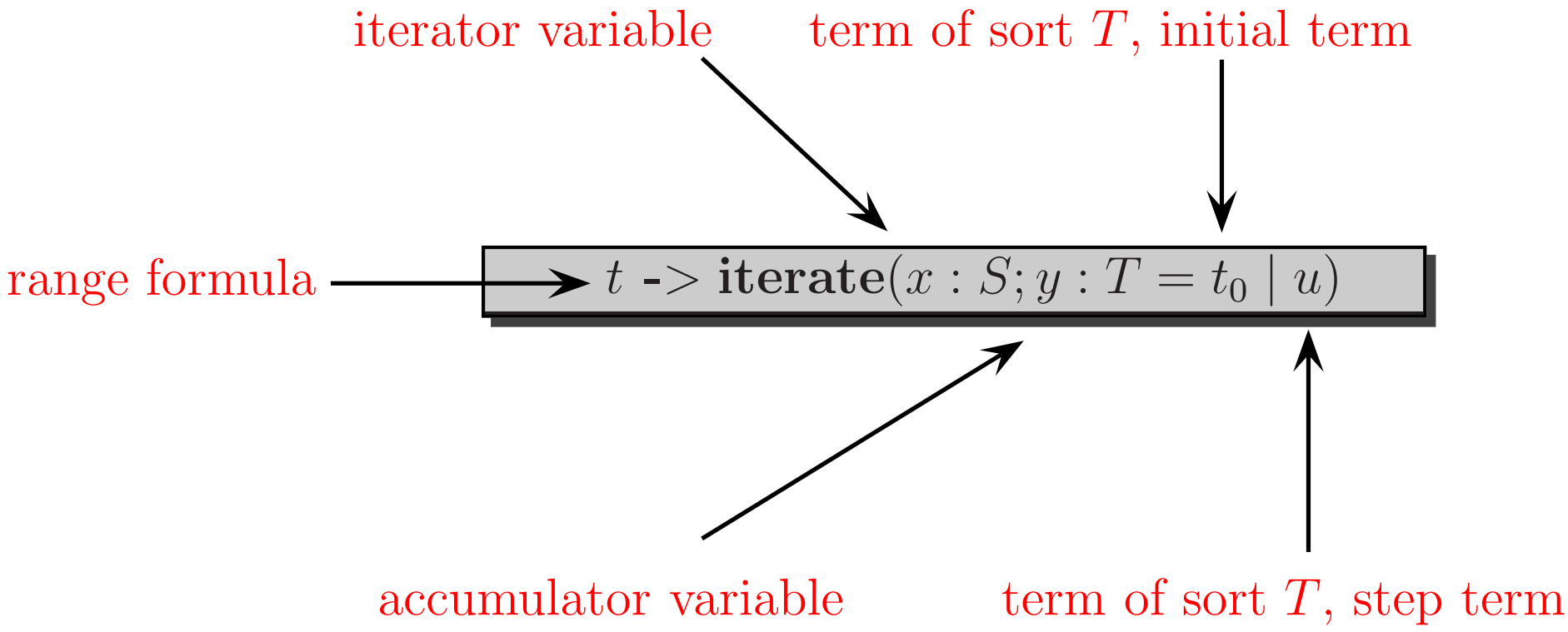
context **p:Papers**

inv **Papers.allInstances –>**

iterate(x:Paper ; y:Int = 0 | y+x.pages)

= Papers.sumpages

Syntax of the *iterate* construct



iterate: Example 1

Adding a new operation *occurences* to the built-in OCL type $String$

string.occurences(string2:String):Set(Integer) **The set of positions in string where an occurence of string2 as a substring starts. Strings start with position** 0**.**

iterate: Example 1

Adding a new operation *occurences* to the built-in OCL type *String*

string.occurences(string2:String):Set(Integer) **The set of positions in string where an occurence of string2 as a substring starts. Strings start with position 0.**

```
pre    : string2.size =< string.size
post   :  result   =   { 0 .. (string.size - string2.size) } ->
                       iterate(x;  y:Set(Integer)=Set{} |
                       if string.substring(x,x+string2.size) = string2
                       then y  -> including(x) else y)
```

iterate: Example 2

Adding a new operation *substringOcc* to the built-in OCL type $String$

string.substringOcc(string2:String):Boolean **True if string2 occurs at least once as a substring in string.**

iterate: Example 2

Adding a new operation *substringOcc* to the built-in OCL type $String$

string.substringOcc(string2:String):Boolean True if string2 occurs at least once as a substring in string.

post : result = (string2.size =< string.size) and not (string.occurences(string2) –> isEmpty)

Quantifiers

$$t \texttt{->} iterate(x; y : Boolean = true \mid y \textbf{ and } a)$$

- t **is an expression of type** $Set(T)$
- x **is a variable of type** T
- a **is an expression of type** $Boolean$

Quantifiers

$$t \texttt{->} iterate(x; y : Boolean = true \mid y \textbf{ and } a)$$

- t **is an expression of type** $Set(T)$
- x **is a variable of type** T
- a **is an expression of type** $Boolean$

Can be equivalently expressed by

$$t \texttt{->} forAll(x \mid a)$$

Quantifiers

$$t \text{->} iterate(x; y : Boolean = true \mid y \textbf{ and } a)$$

- t **is an expression of type** $Set(T)$
- x **is a variable of type** T
- a **is an expression of type** $Boolean$

Can be equivalently expressed by

$$t \text{->} forAll(x \mid a)$$

Likewise

$$t \text{->} iterate(x; y : Boolean = false \mid y \textbf{ or } a)$$

Quantifiers

$$t \text{->} iterate(x; y : Boolean = true \mid y \textbf{ and } a)$$

- t **is an expression of type** $Set(T)$
- x **is a variable of type** T
- a **is an expression of type** $Boolean$

Can be equivalently expressed by

$$t \text{->} forAll(x \mid a)$$

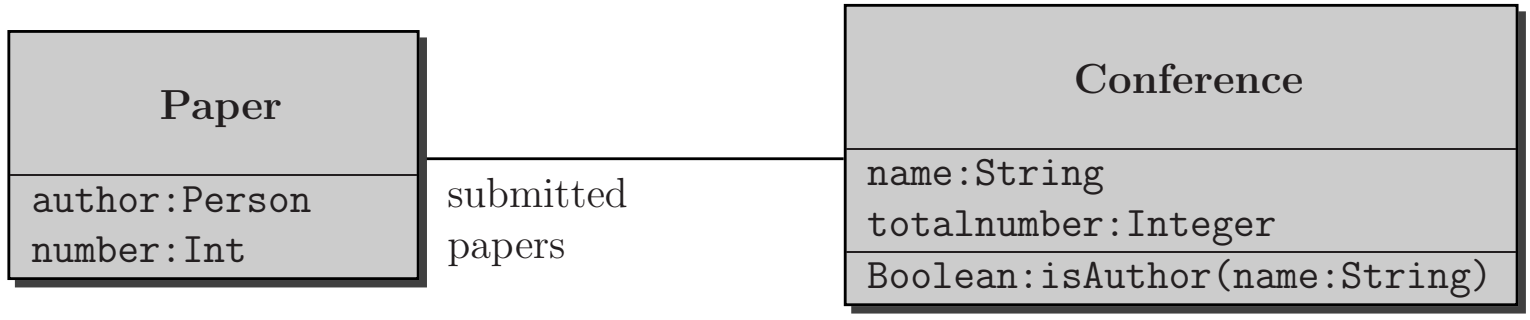
Likewise

$$t \text{->} iterate(x; y : Boolean = false \mid y \textbf{ or } a)$$

Can be expressed by

$$t \text{->} exists(x \mid a)$$

Collecting Elements



context c**:Conference::isAuthor(name:String)**

pre **true**

post **result =**
c.sp –> collect(p | p.author.name) –> includes(name)

Reducing *collect* to *iterate*

set –> collect(x | expr) : Bag(T)

=

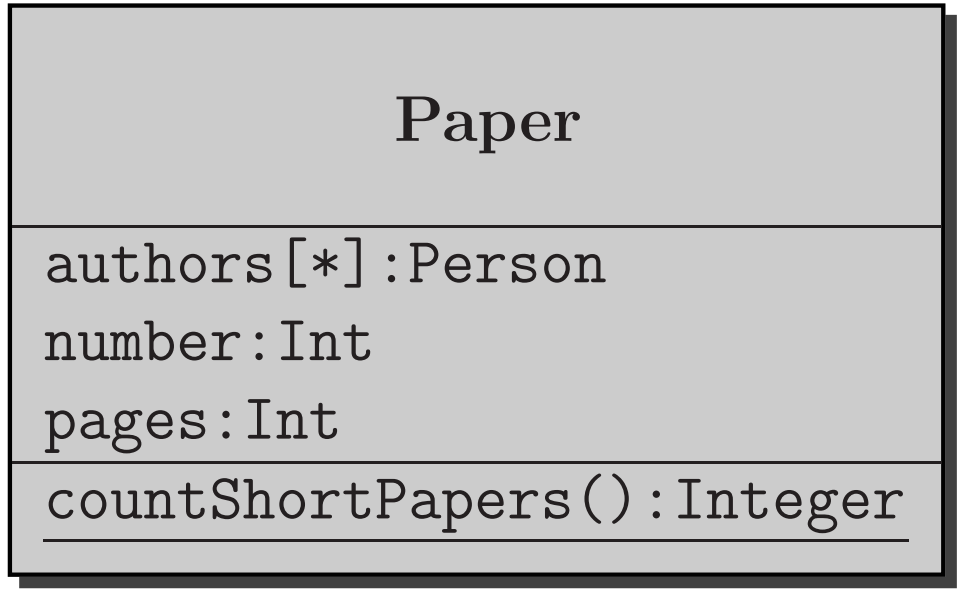
set –> iterate(x; acc : Bag(T) = Bag{} | acc –> including(expr))

Evaluation of

c.sp –> collect(p | p.authors.name)

involves implicit flattening.

Selecting Elements



context Paper::countShortPapers():Integer

pre true

post result =
Paper.allInstances –>
select(p | p.pages < 10) –> size

Reducing *select* to *iterate*

s –> select(x | expr) : Set(T) =

s –> iterate(x; acc : Set(T) = Set{} |

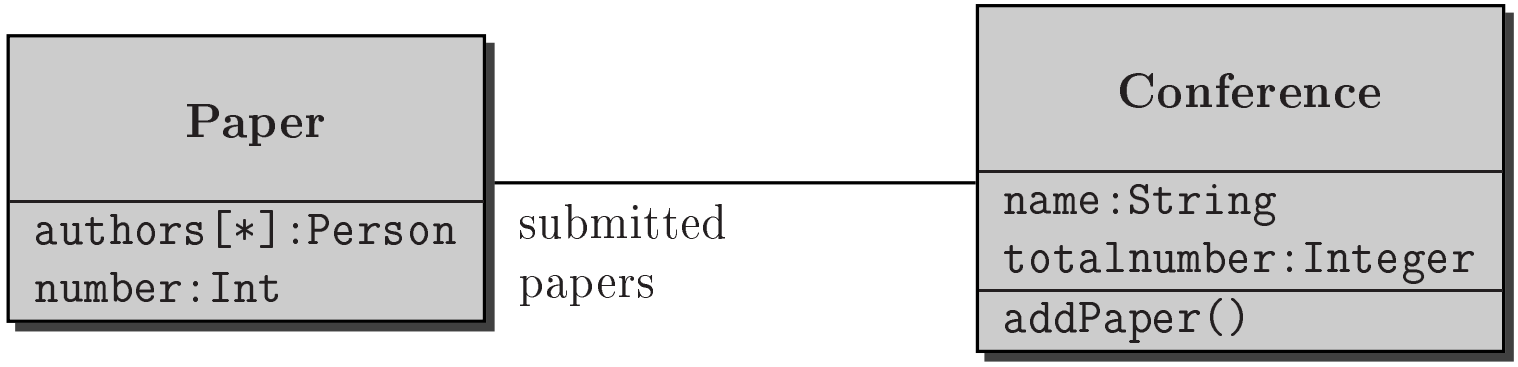
if expr then acc –> including(x)

else acc)

where

- s **is of type** $Set(T)$
- ***expr* is an OCL expression of type *Boolean***

Refering to Previous Values

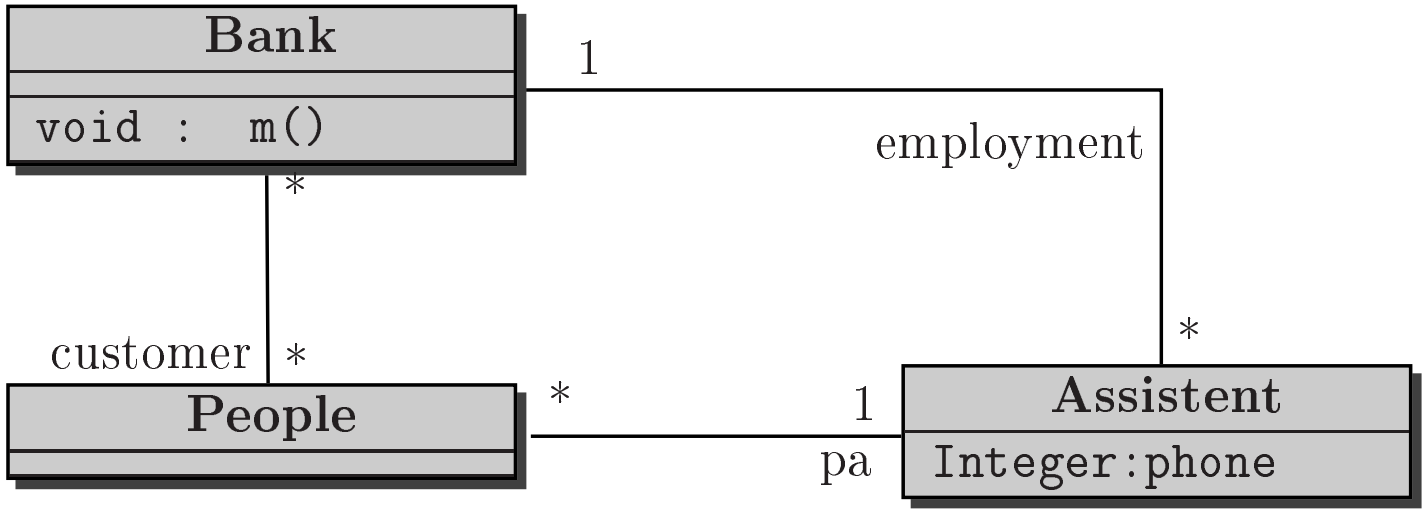


context c**:Conference::addPaper()**

pre **true**

post **totalnumber = totalnumber@pre + 1**

Multiple Occurences of @pre



$c.pa.phone$	**the new phone number of the current p.a.**
$c.pa@pre.phone$	**the new phone number of the previous p.a.**
$c.pa.phone@pre$	**the old phone number of the current p.a.**
$c.pa@pre.phone@pre$	**the old phone number of the previous p.a.**