SWEN 223

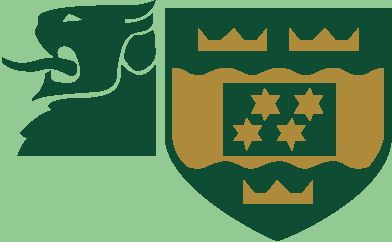
Software Engineering Analysis

Object-Constraint

Language

Thomas Kuhne

Victoria University of Wellington  
[Thomas.Kuehne@ecs.vuw.ac.nz](mailto:Thomas.Kuehne@ecs.vuw.ac.nz), Ext. 5443, Room Cotton 233



Achieving Well-Formedness

* UML class diagrams are type models for all their possible instance models

» concepts, their allowed relationships and multiplicities restrict the set of instance models conforming to a type model

* However, a number of restrictions cannot be expressed by means of the visual notation only

» Similar in programming languages: The grammar has to be augmented by static semantics rules

1 r\

**dMS**

13/05/2016 % f y

Applications

Constraints are used to express, e.g.,

* Limits

» constrain values to certain ranges

* Uniqueness

» constrain instances values to be unique

* Consistency

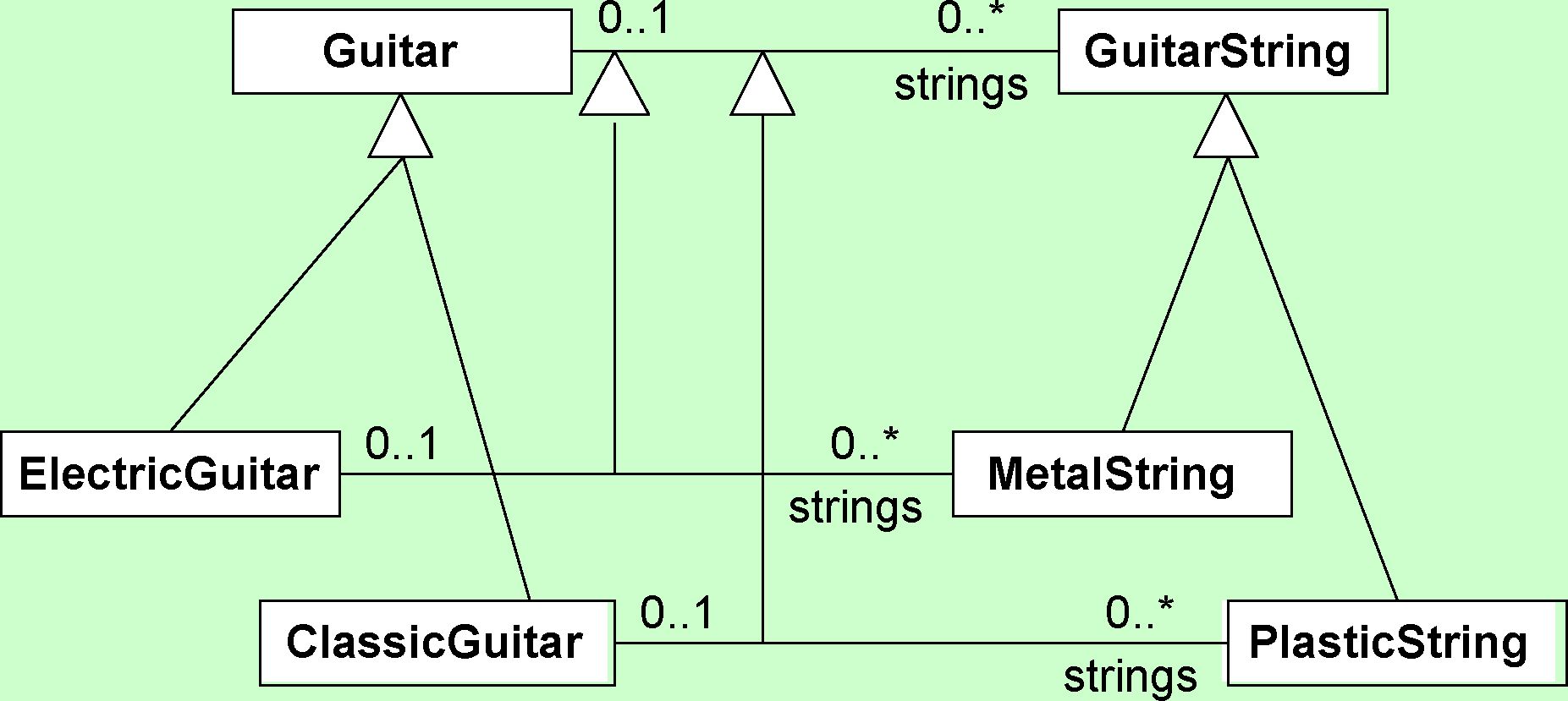
» express invariants on data structures

* Contracts

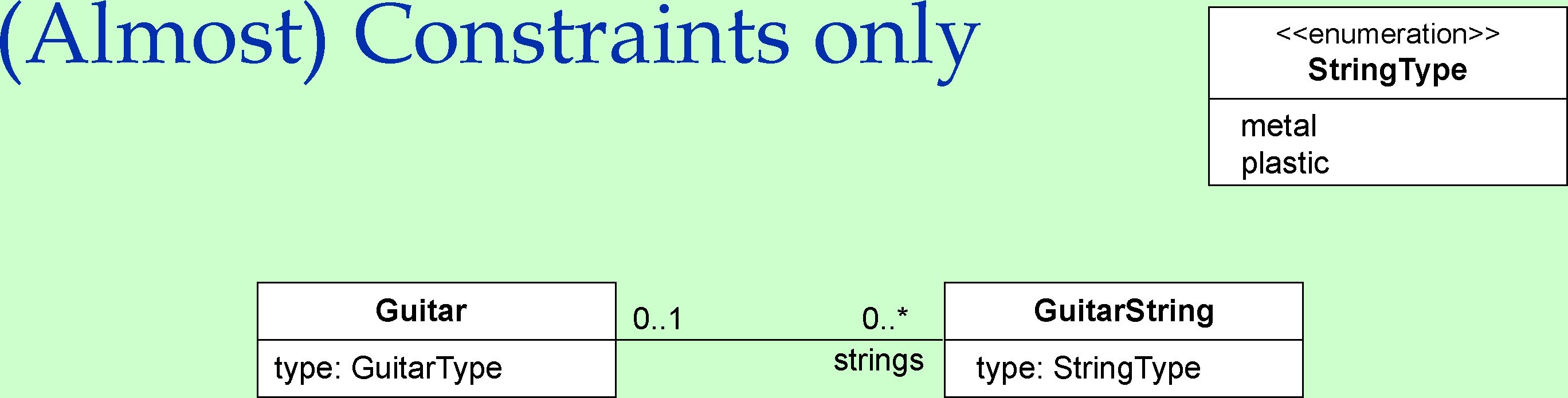
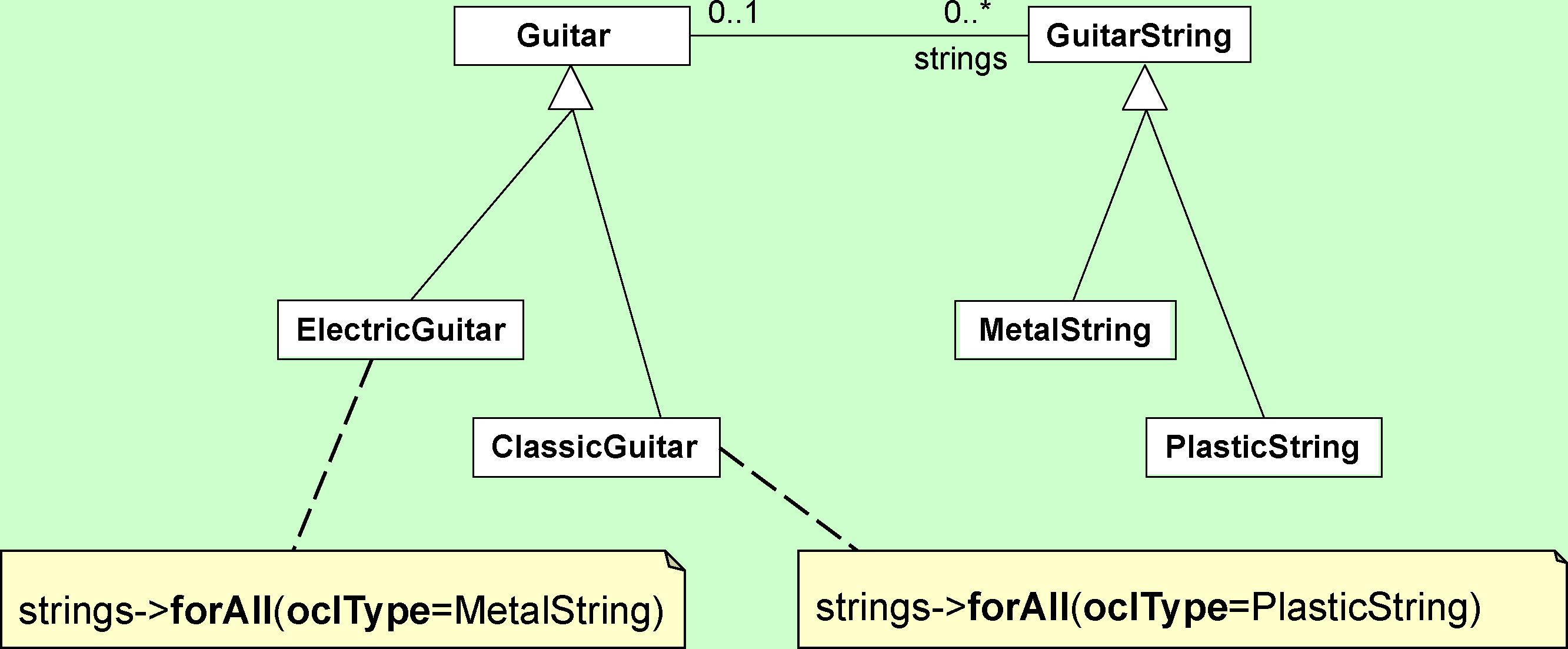
» pre- and post conditions for operations

1 r\

Fully graphical



Graphical with constraints



; ^

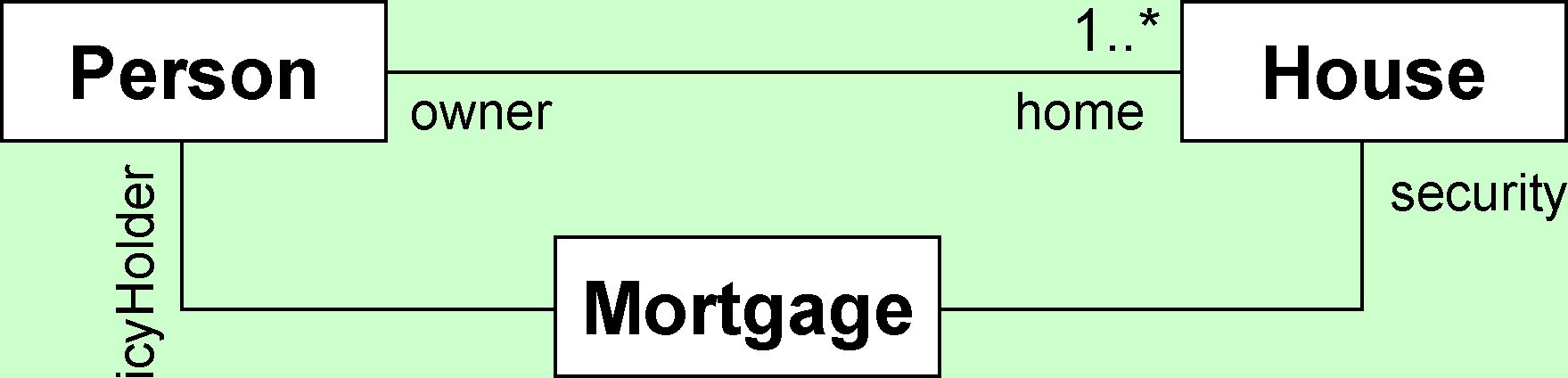
context Guitar inv correctStrings:

(self.type = GuitarType::electric implies strings->forAll(type = StringType::metal) and (self.type = GuitarType::classic implies strings->forAll(type = StringType::plastic)



Beyond Types

Fixing Identity



o

Q.

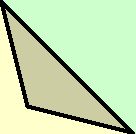
context Mortgage inv ownsSecurity: seIf.policyHolder = self.security.owner



Invariants

|  |  |  |
| --- | --- | --- |
| Person | \*  > | Job |
| isUnemployed : Boolean |  | title : String |
| income : Integer |  | salary : Integer |

context Person inv appropriateSalary: let income : Integer = self.job.salary->sum() in let hasTitle(t : String) : Boolean = self.job->exists(title = t) in if isUnemployed then self.income < 300 else



self.hasTitle(‘manager’) implies self.income >= 4000 endif



* Invariant

» constraint that states a condition that must always be met by all instances of the type. Invariants must be true all the time (except during operation execution).

* Precondition

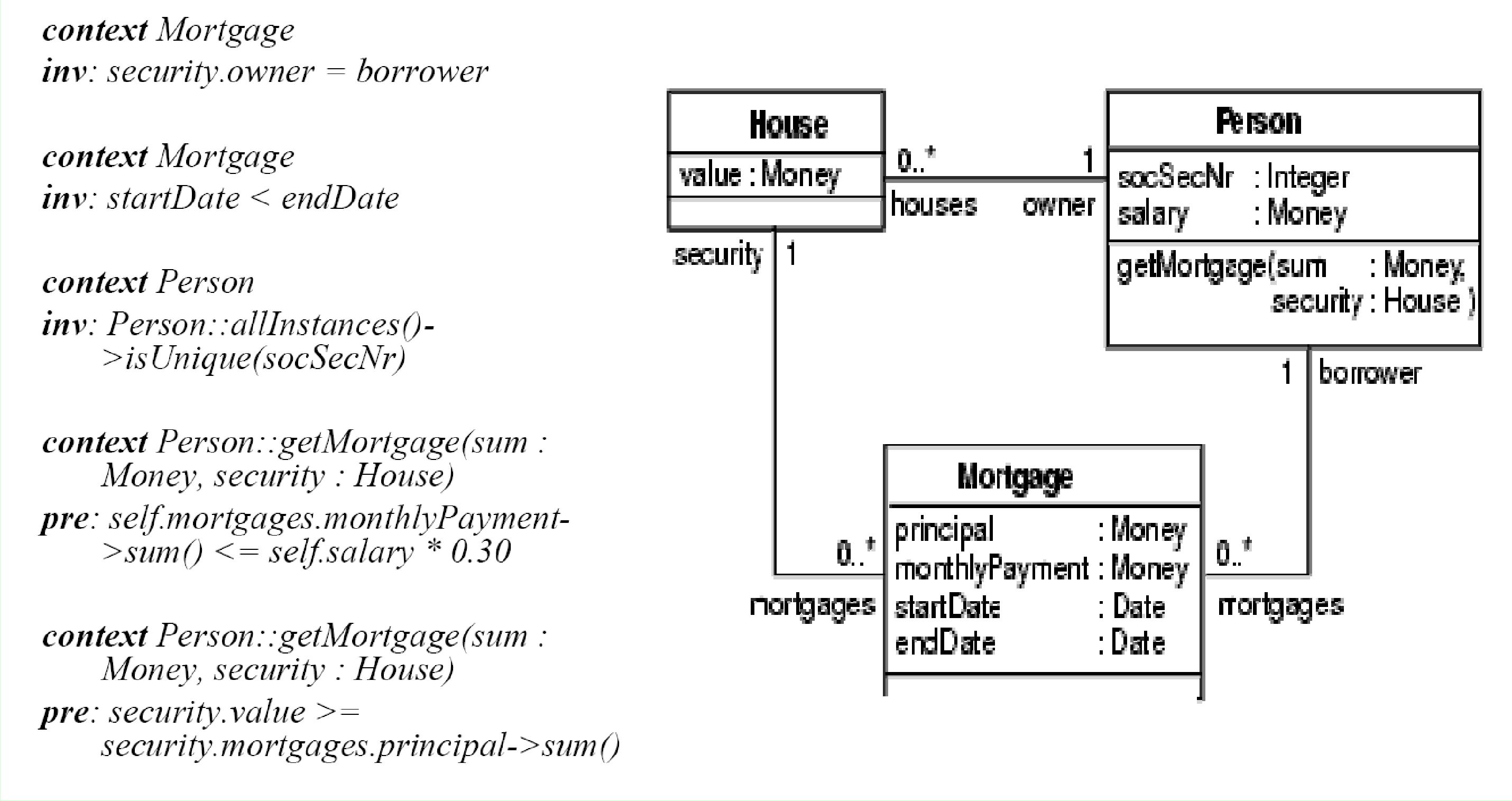
» a precondition to an operation is a restriction that must be true before the operation is going to be executed.

* Postcondition

» a postcondition to an operation is a restriction that must be true after that the operation has just ended its execution.



Beyond Types



Operations on Sets

***Descr****i****p****ti****on***

***Operation***

*s->intersection(t)*

*s->union(t)*

*s->notEmpty()*

*s->size()*

*s->excludes(o)*

*s->isEmpty()*

*s->excludesAll(u)*

*s->includesAll(u)*

*s->includes(o)*

*s->count(o)*

Returns the intersection of sets s and t Computers the union of s and t True if s contains at least one element Returns the number of elements in set s True if o is not an element of s True if s doesn’t contain any elements True if all elements of set u are not in s True if all elements of set u are in s True if o is an element of s Number of times element o occurs in s



Iterators over Sets

|  |  |
| --- | --- |
| Operation  s->reject(expr) | Doscniption  Returns a subset of s containing all elements for which expr is false |
| s->select(expr) | Returns a subset of s containing all elements for which expr is true |
| s->forAll(expr) | Returns true if expr is true for all elements in the source collection |
| s->exists(expr) | Returns true if there is at least one element in the source collection for which expr is true |

s->collect(expr) Returns the set of objects that result from evaluating

|  |  |
| --- | --- |
| s->any(expr) | expr for each element in the source collection  Returns a random element for which expr is true |
|  | 1 ^ ^ |

OCL vs Alloy

* OCL

» integrated into the UML

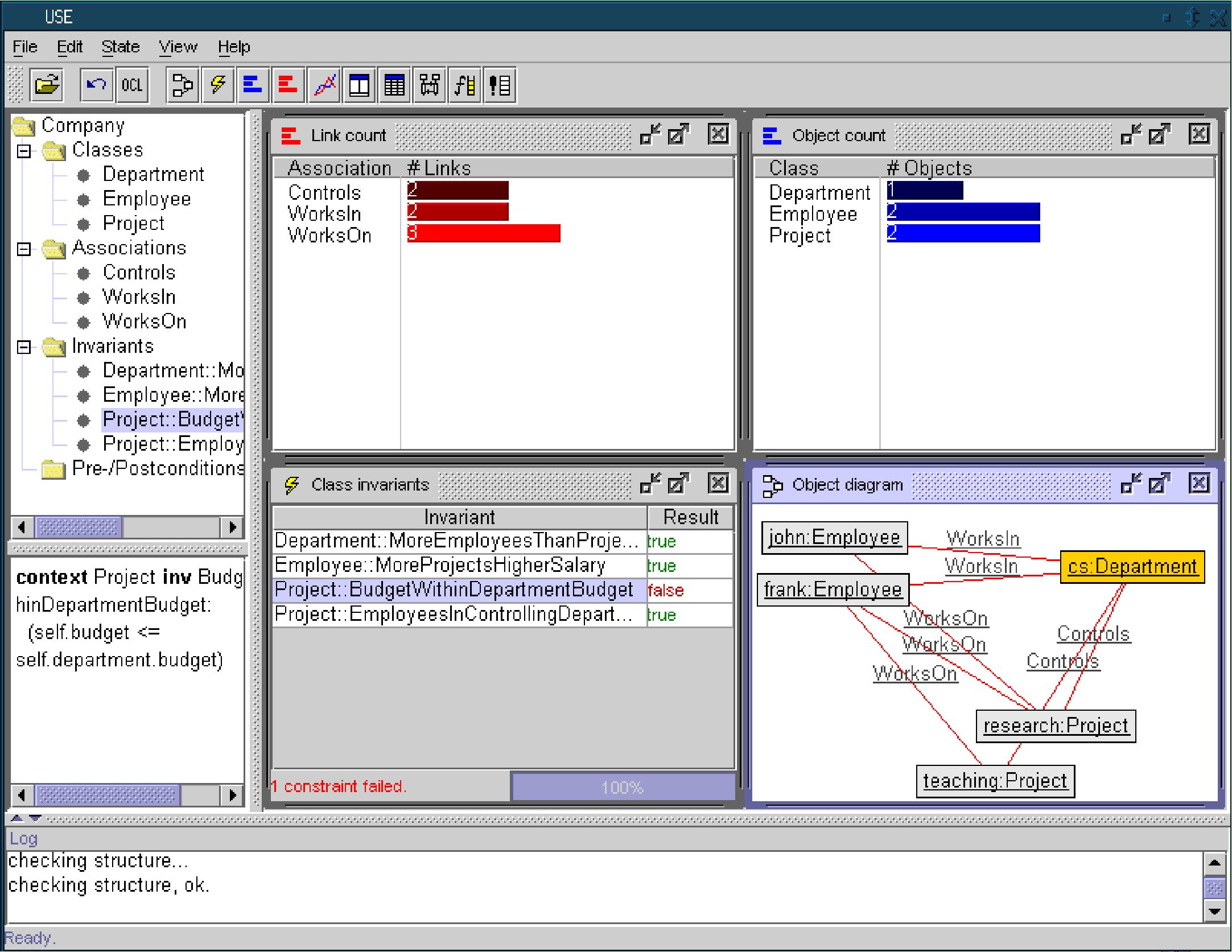
* each UML class/interface is automatically an OCL type
* navigation along associations » fully supports primitive types » allows recursive definitions
* Alloy

» supports several styles of specification — OO, relational, first order logic » comes with a solver

—i ***<r°\***

USE Tool

System states can be created & manipulated



For each snapshot the constraints are auto­matically checked

Language Features

* Textual Notation

» supposedly easier to read than standard logic notations

* Declarative

» expressions have no side effects

» “loose semantics” allows admissible solutions but does not prescribe specific solutions

* Statically Typed

» type errors may be caught before evaluation



Further Resources for OCL

• The Object Constraint Language

» ISBN 0-201-37940-6 (old)

» ISBN 0-321-17936-6 (newer, UML 2.0 + MDA)

• OCL home page

[» www.klasse.nl/ocl/index.htm](http://www.klasse.nl/ocl/index.htm) » <http://www.klasse.nl/books/ocl-intro.html>

JOS WARMER ANNEKE KEEPPE



**The Object Constraint Language**

Precise Modeling with UML

