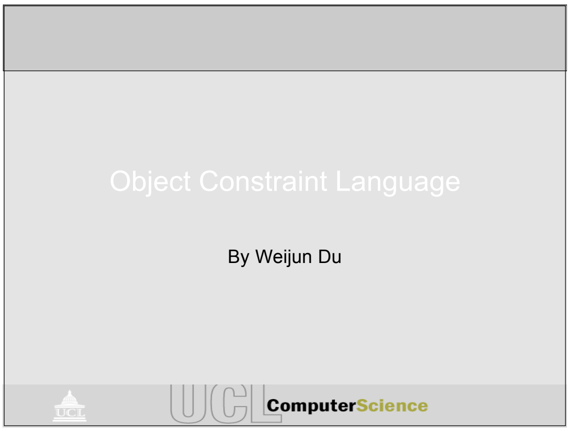
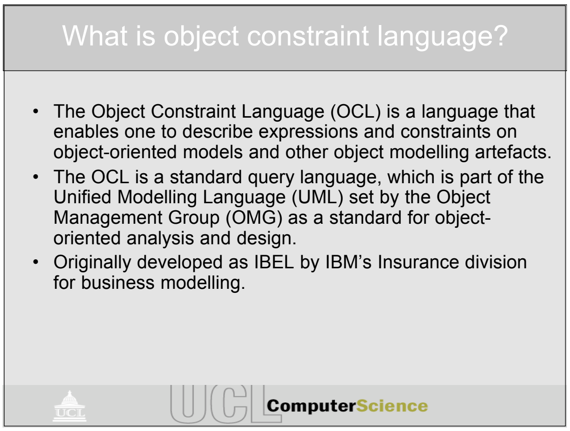


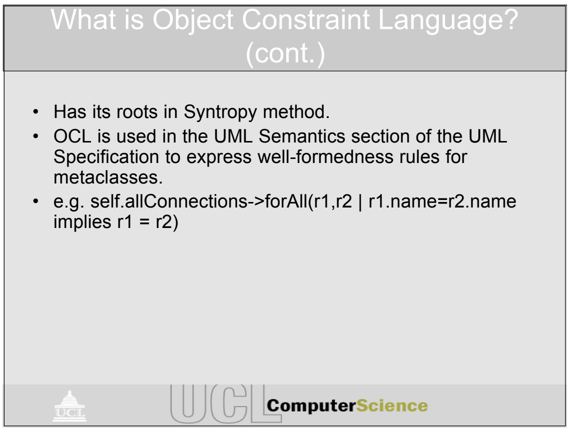
Object Constraint Language

By Weijun Du



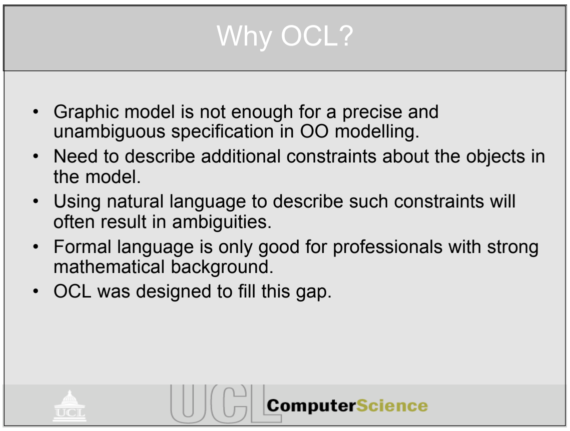
What is object constraint language?

- The Object Constraint Language (OCL) is a language that enables one to describe expressions and constraints on object-oriented models and other object modelling artefacts.
- The OCL is a standard query language, which is part of the Unified Modelling Language (UML) set by the Object Management Group (OMG) as a standard for object-oriented analysis and design.
- Originally developed as IBEL by IBM's Insurance division for business modelling.



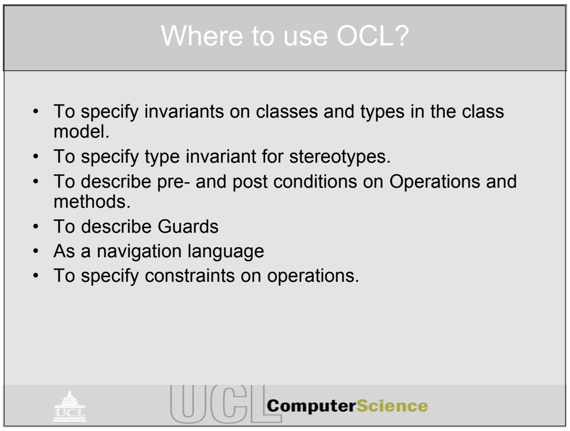
What is Object Constraint Language? (cont.)

- Has its roots in Syntropy method.
- OCL is used in the UML Semantics section of the UML Specification to express well-formedness rules for metaclasses.
- e.g. self.allConnections->forAll(r1,r2 | r1.name=r2.name implies r1 = r2)



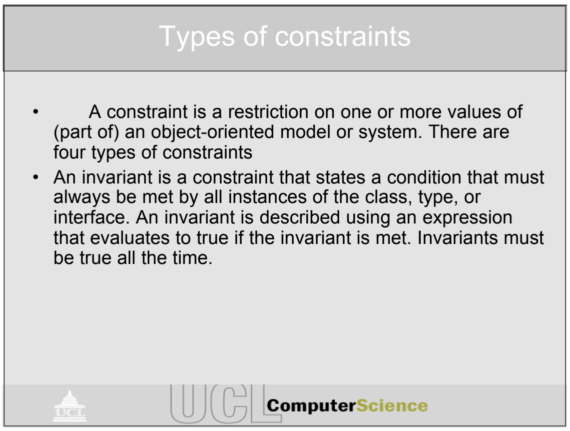
Why OCL?

- Graphic model is not enough for a precise and unambiguous specification in OO modelling.
- Need to describe additional constraints about the objects in the model.
- Using natural language to describe such constraints will often result in ambiguities.
- Formal language is only good for professionals with strong mathematical background.
- OCL was designed to fill this gap.



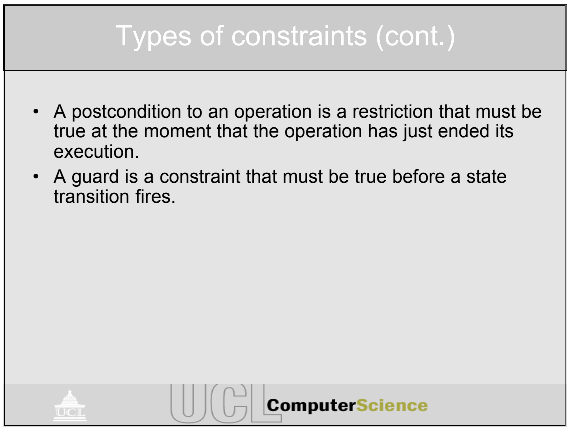
Where to use OCL?

- To specify invariants on classes and types in the class model.
- To specify type invariant for stereotypes.
- To describe pre- and post conditions on Operations and methods.
- To describe Guards
- As a navigation language
- To specify constraints on operations.



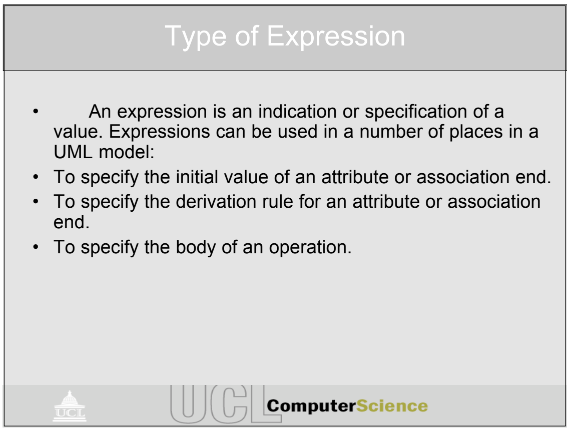
Types of constraints

- A constraint is a restriction on one or more values of (part of) an object-oriented model or system. There are four types of constraints
- An invariant is a constraint that states a condition that must always be met by all instances of the class, type, or interface. An invariant is described using an expression that evaluates to true if the invariant is met. Invariants must be true all the time.



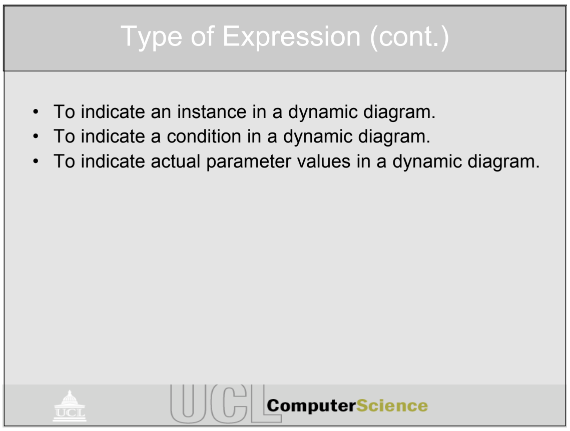
Types of constraints (cont.)

- A postcondition to an operation is a restriction that must be true at the moment that the operation has just ended its execution.
- A guard is a constraint that must be true before a state transition fires.



Type of Expression

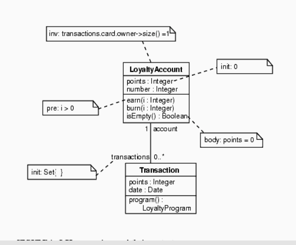
- An expression is an indication or specification of a value. Expressions can be used in a number of places in a UML model:
- To specify the initial value of an attribute or association end.
- To specify the derivation rule for an attribute or association end.
- To specify the body of an operation.



Type of Expression (cont.)

- To indicate an instance in a dynamic diagram.
- To indicate a condition in a dynamic diagram.
- To indicate actual parameter values in a dynamic diagram.

Example of Class Diagram

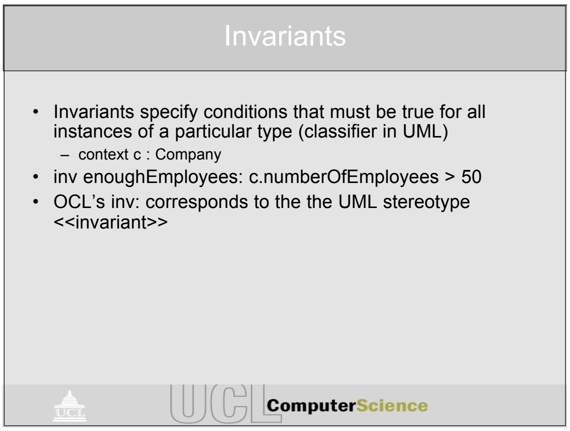


OCL and the UML Metamodel

- Each OCL expression is evaluated in the context of an instance of a specific type (class)
  - this context is called self
  - if the context is clear then the keyword self can be omitted
- If the context is not clear then it can be specified at the start of an expression
- context TypeName

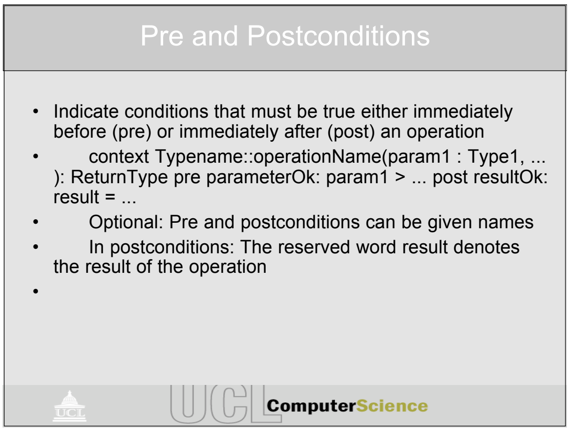
Self

- Each OCL expression is written in the context of an instance of a specific type. In an OCL expression, the name self is used to refer to the contextual instance.



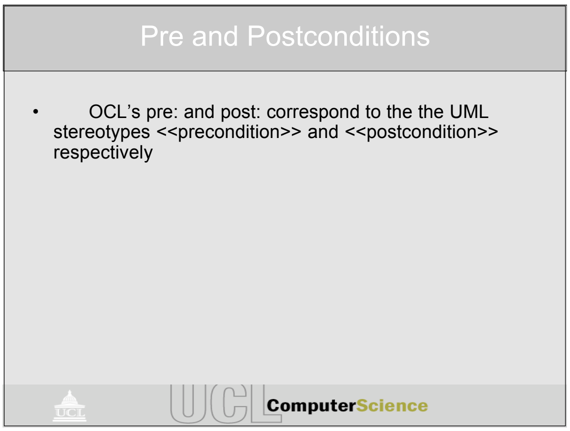
Invariants

- Invariants specify conditions that must be true for all instances of a particular type (classifier in UML)
  - context c : Company
- inv enoughEmployees: c.numberOfEmployees > 50
- OCL’s inv: corresponds to the the UML stereotype <<invariant>>



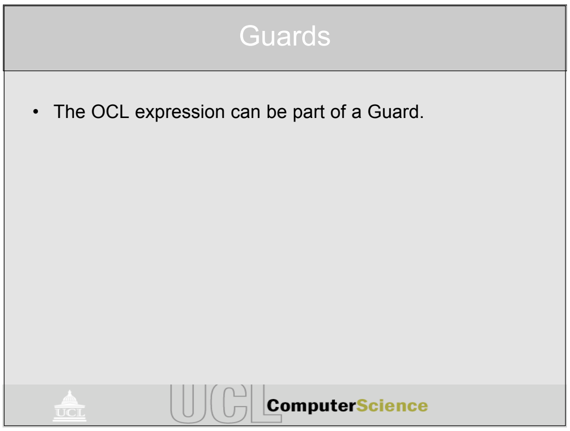
Pre and Postconditions

- Indicate conditions that must be true either immediately before (pre) or immediately after (post) an operation
- context Typename::operationName(param1 : Type1, ...): ReturnType pre parameterOk: param1 > ... post resultOk: result = ...
- Optional: Pre and postconditions can be given names
- In postconditions: The reserved word result denotes the result of the operation



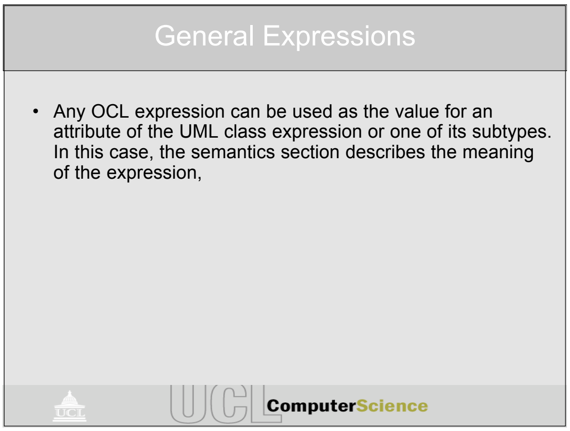
Pre and Postconditions

- OCL’s pre: and post: correspond to the the UML stereotypes <<precondition>> and <<postcondition>> respectively



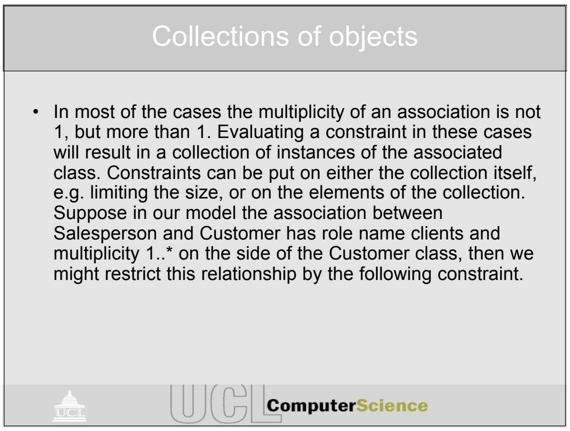
Guards

- The OCL expression can be part of a Guard.



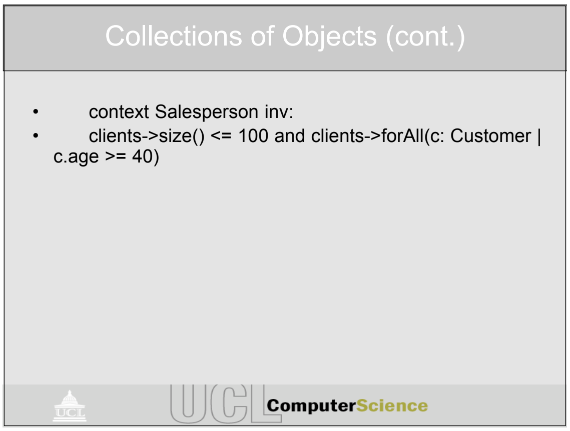
General Expressions

- Any OCL expression can be used as the value for an attribute of the UML class expression or one of its subtypes. In this case, the semantics section describes the meaning of the expression,



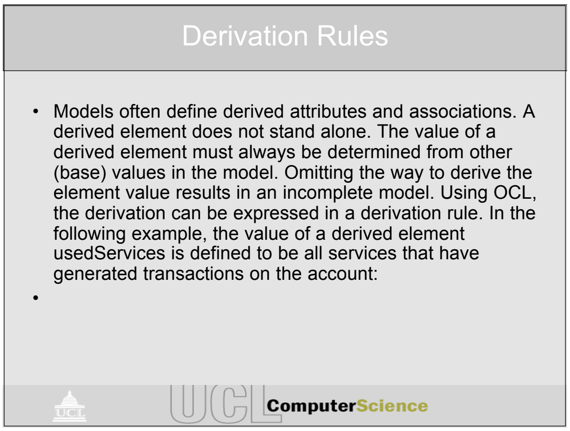
Collections of objects

- In most of the cases the multiplicity of an association is not 1, but more than 1. Evaluating a constraint in these cases will result in a collection of instances of the associated class. Constraints can be put on either the collection itself, e.g. limiting the size, or on the elements of the collection. Suppose in our model the association between Salesperson and Customer has role name clients and multiplicity 1..* on the side of the Customer class, then we might restrict this relationship by the following constraint.



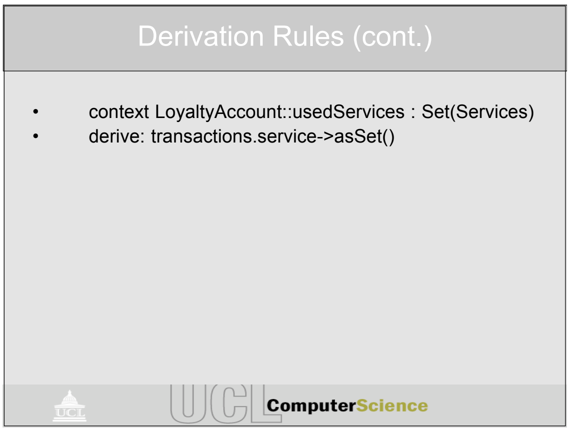
Collections of Objects (cont.)

- context Salesperson inv:
- clients->size() <= 100 and clients->forAll(c: Customer | c.age >= 40)



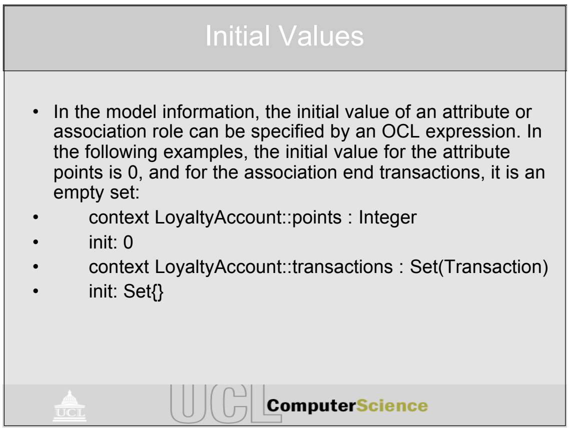
Derivation Rules

- Models often define derived attributes and associations. A derived element does not stand alone. The value of a derived element must always be determined from other (base) values in the model. Omitting the way to derive the element value results in an incomplete model. Using OCL, the derivation can be expressed in a derivation rule. In the following example, the value of a derived element usedServices is defined to be all services that have generated transactions on the account:
-



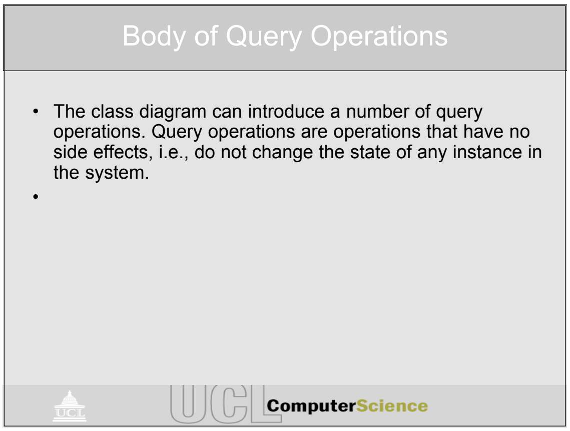
Derivation Rules (cont.)

- context LoyaltyAccount::usedServices : Set(Services)
- derive: transactions.service->asSet()



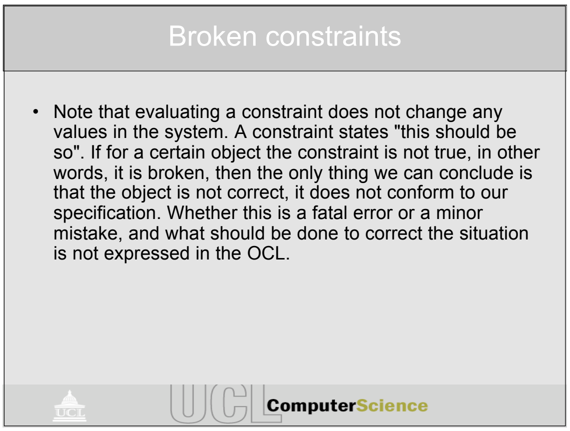
Initial Values

- In the model information, the initial value of an attribute or association role can be specified by an OCL expression. In the following examples, the initial value for the attribute points is 0, and for the association end transactions, it is an empty set:
- context LoyaltyAccount::points : Integer
- init: 0
- context LoyaltyAccount::transactions : Set(Transaction)
- init: Set{}



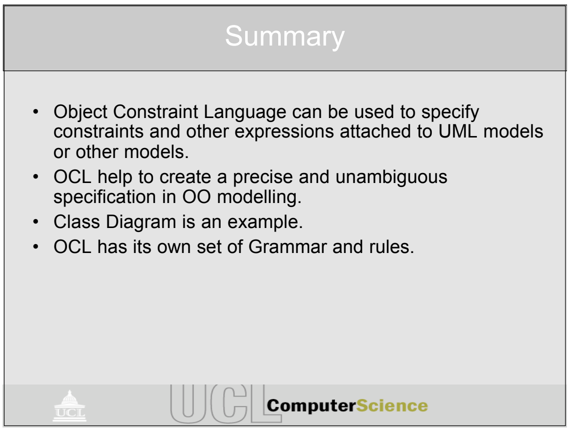
Body of Query Operations

- The class diagram can introduce a number of query operations. Query operations are operations that have no side effects, i.e., do not change the state of any instance in the system.



Broken constraints

- Note that evaluating a constraint does not change any values in the system. A constraint states "this should be so". If for a certain object the constraint is not true, in other words, it is broken, then the only thing we can conclude is that the object is not correct, it does not conform to our specification. Whether this is a fatal error or a minor mistake, and what should be done to correct the situation is not expressed in the OCL.



Summary

- Object Constraint Language can be used to specify constraints and other expressions attached to UML models or other models.
- OCL help to create a precise and unambiguous specification in OO modelling.
- Class Diagram is an example.
- OCL has its own set of Grammar and rules.