CS Department in Dresden

OUTPUT Demo Day of the Department

REWERSE© reasoning on the web

Collaboration-Based Language Composition

or: How to model a newspaper-reading sausage-buying grandfather

Uwe Aßmann
Christian Wende
Technische Universität Dresden
Chair of Software Engineering

Fiction or Reality?!

```
use Modula.2.0 for base
use C++.2040 for parameterization and initialization
use SQL.5.0 for data-fetch
use BETA for slots


template class S, DB {
  IMPLEMENTATION MODULE WebServer<S>;
    PROCEDURE <<..>> END;
  BEGIN
    S: servletGenerator = DB.init;
    R: relation = select all from DB
                        where Person == "Assmann";
  END
}
```

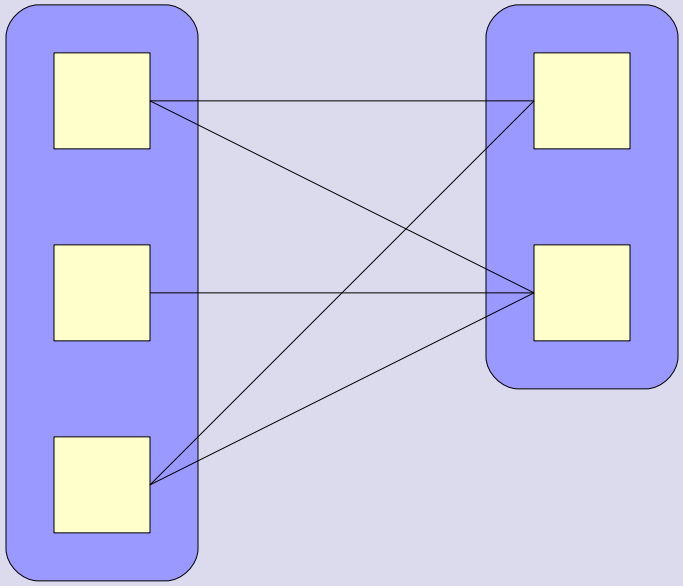
Composition of Languages

Problem:

- Composition of two languages
- Extension of a base language
- Specification of a crosscut in the semantics
  - Hand-invasive edits

Traditionally done with declarative specifications

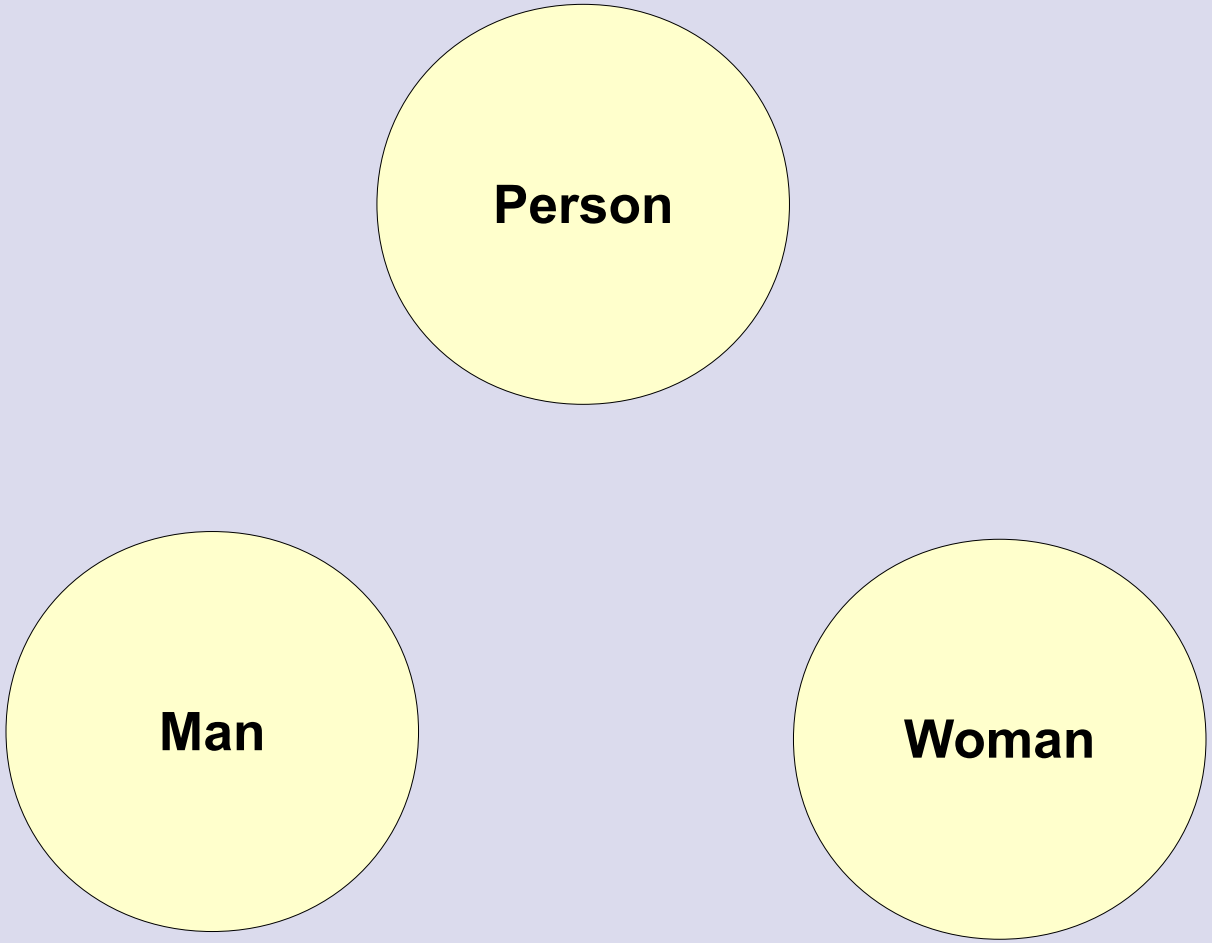
- Composition of Attribute grammars
  - ELI, fnc-2, LISA, Silver, JastAdd
- Composition of Natural Semantics
  - Typol, RML
- Logic
  - Datalog, OWL



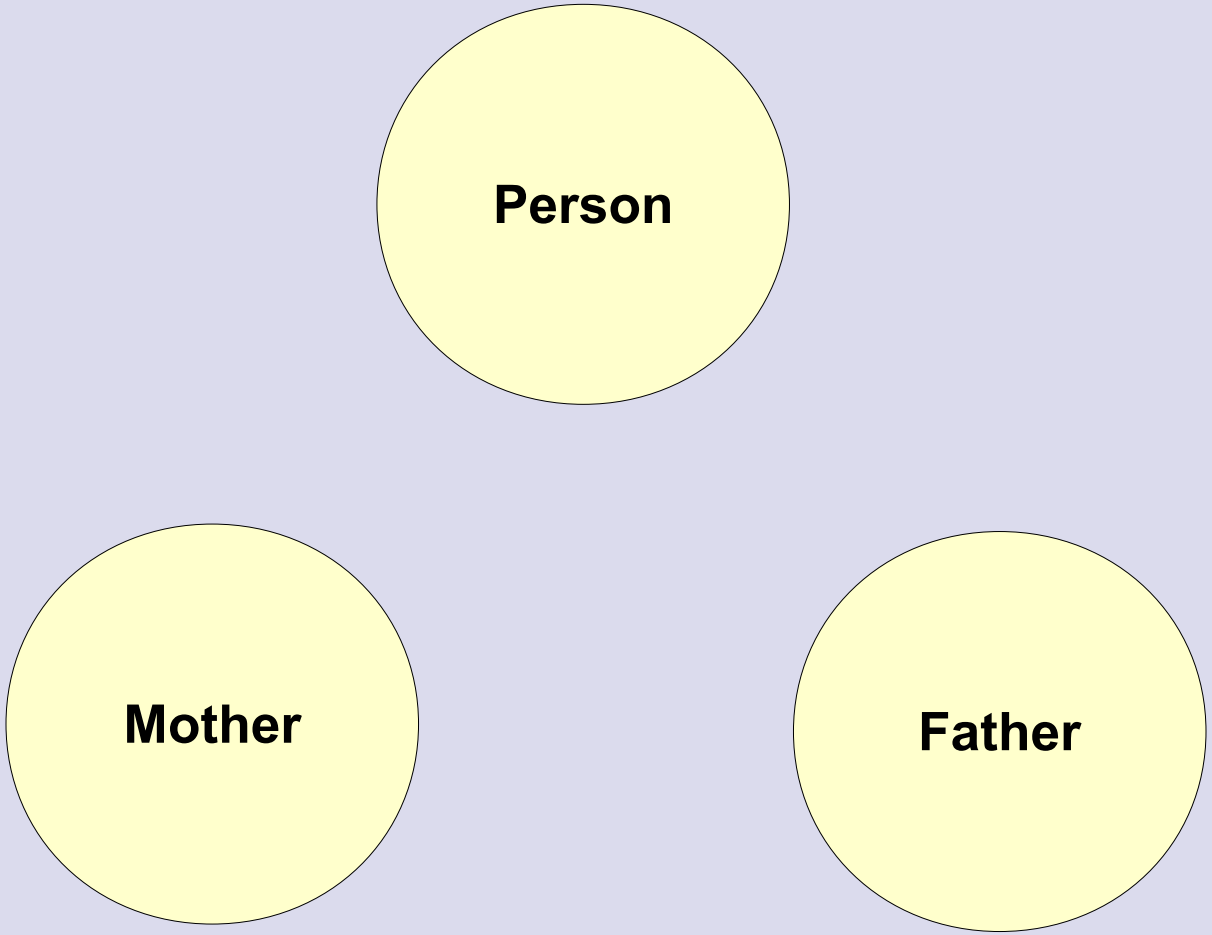
Goal: Simplicity, Automation

Let's look at modern O-O!

A Riddle..



Another Riddle..



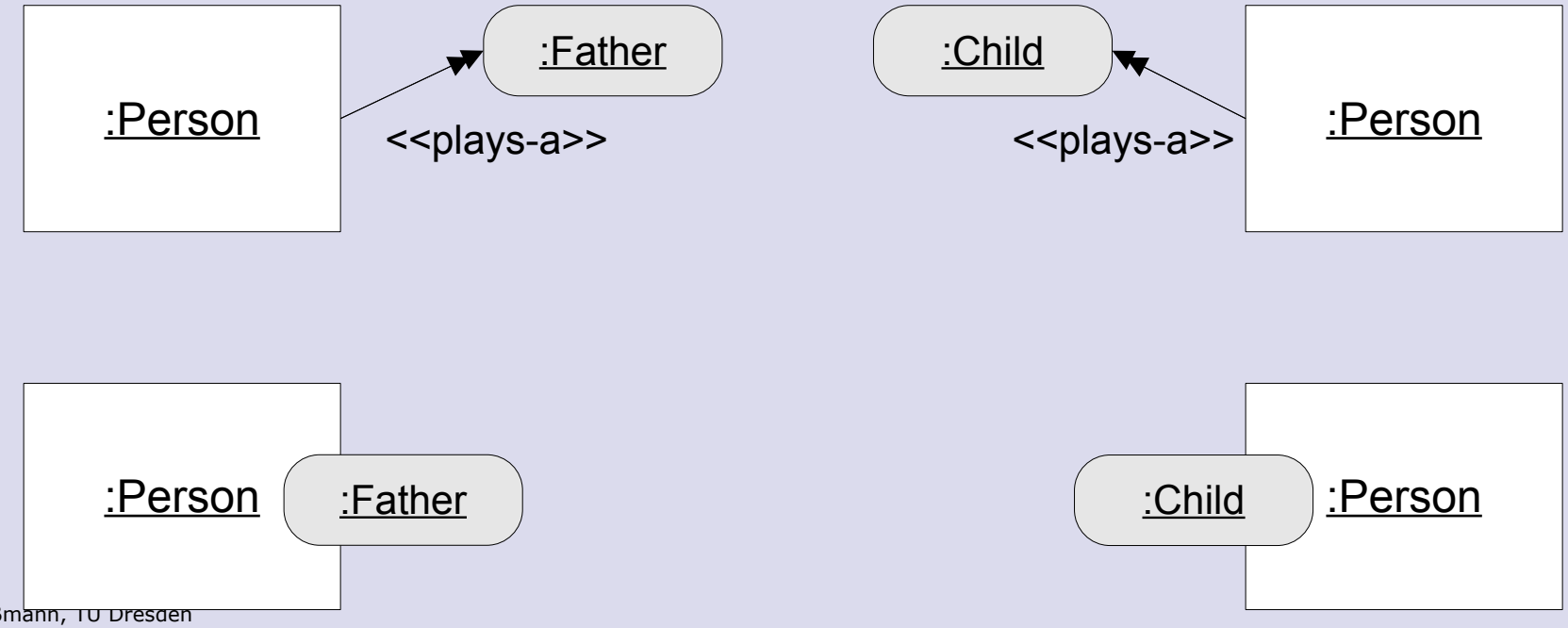
New Developments in Object-Oriented Modeling

Collaboration-Based Modeling (Role Modeling)

Databases [Bachmann]

Factorization [Steimann]

Research in Design Patterns [Reenskaug, Riehle/Gross]



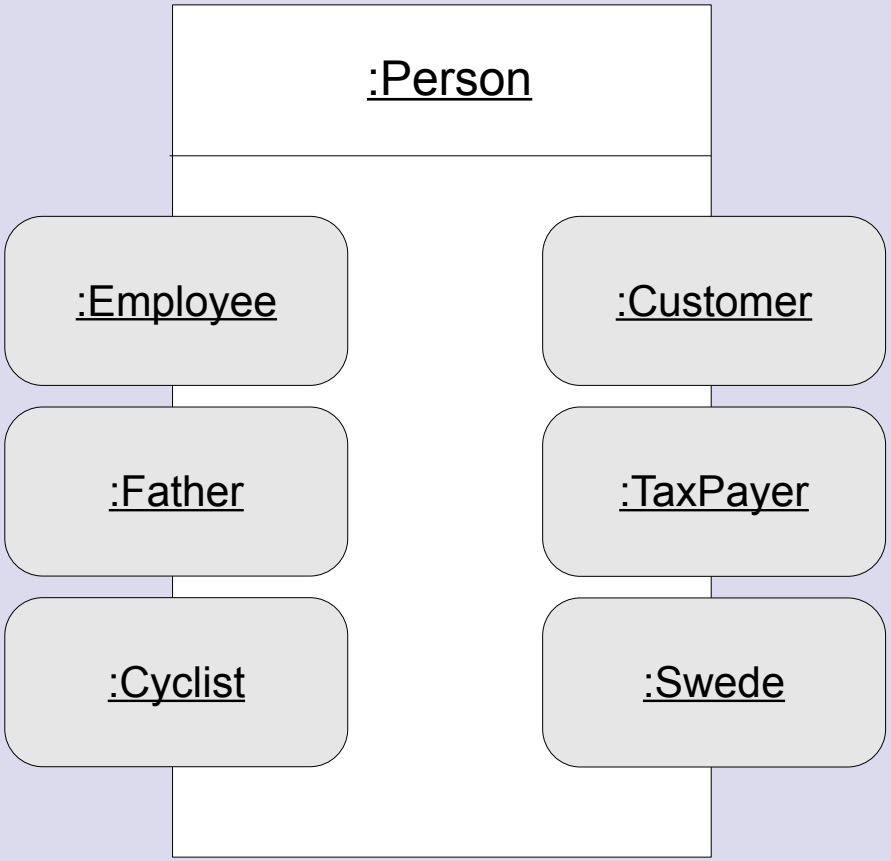
What are Roles?

A *role* is a *dynamic view* onto an object

- Roles are *played* by the objects (the object is the *player* of the role)
- A *partial object*

Roles are tied to *collaborations*

- Do not exist standalone, depend on a partner



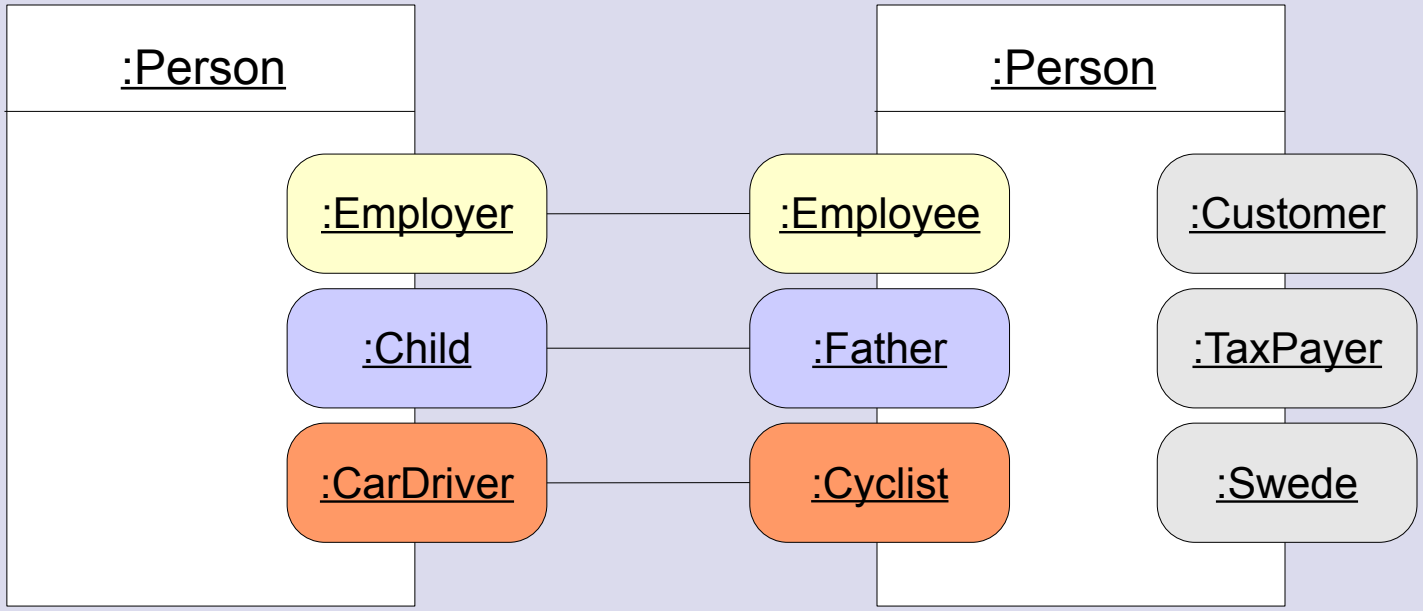
What are Roles?

Roles are *services* of an object in a context

- Roles can be connected to each other
- A role has an *interface*

Roles form *role models,* capturing an area of *concern* [Reenskaug]

- Role models are *collaborative aspects*



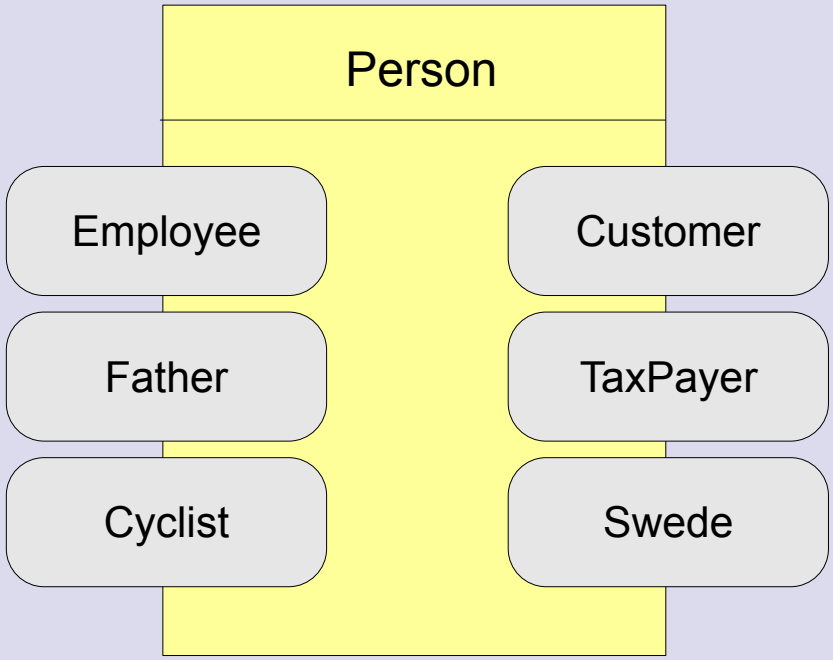
What are Role Types?

Role types (abilities) are

- *service types*
- *dynamic types*
- *collaborative types*

Problem:

- The word "role" is also used on the class level, i.e., for a "role type"

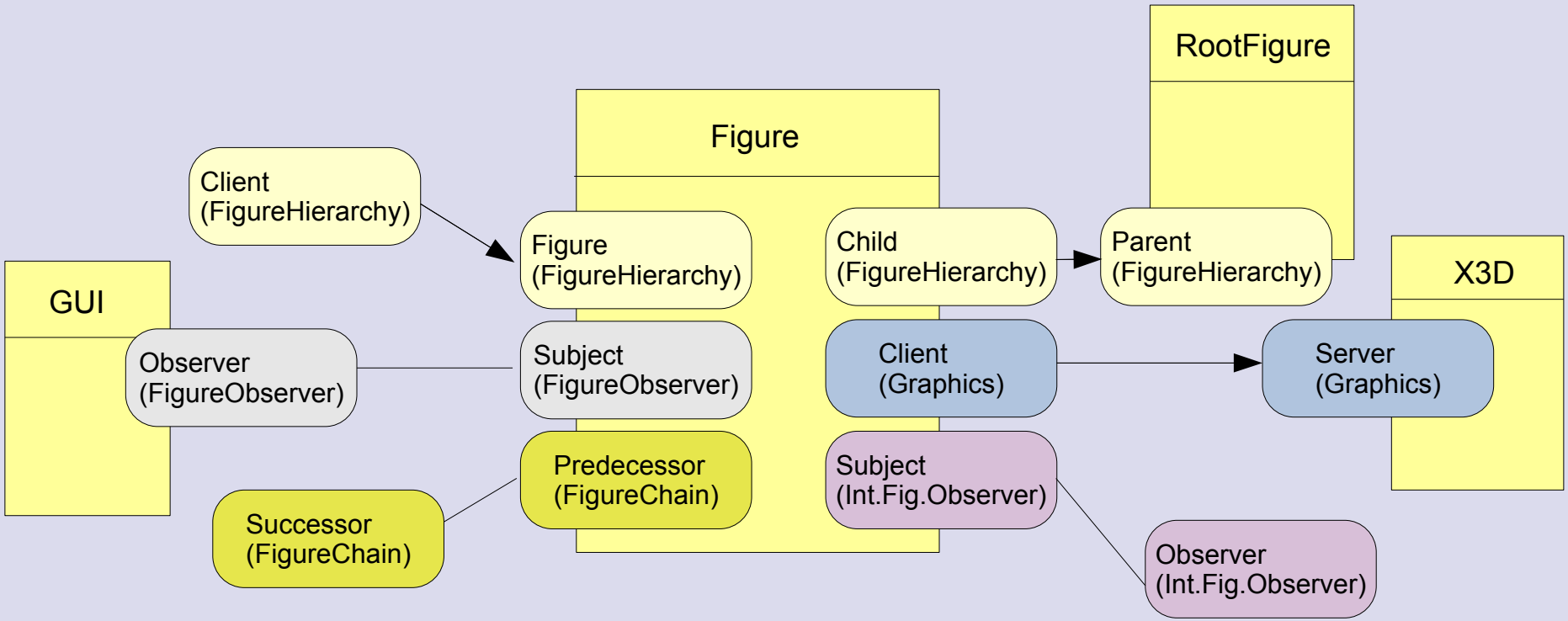


Collaboration Schemas (Role-Type Model)

Collaboration schema (role type model, ability model):

- Set of object collaborations abstracted by a set of role types
- A constraint specification for classes and object collaborations

Ex: A figure can play many roles in different *collaboration schemas*



Role- and Role-Type Models Underly Many Gray-Box Component Models

Views

- Hyperspace (MDSOC)

Collaborative Aspects

- ObjectTeams
- CaesarJ

Template-based languages

- BETA, Yggdrasil
- Invasive Software Composition

The Steimann Factorization

Splitting a type into a tuple of natural and founded parts

Rigid Types [Guarini]

If an object that has a *rigid* type, it cannot stop being of the type without loosing its identity

Example:

- Book is a rigid type
- Reader is a non-rigid type
- Reader can stop reading, but Book stays Book

Rigid types are *tied to the identity* of objects

- A *non-rigid type* is a dynamic type that is indicating a state of the object

Founded and Natural Types

A *founded type (relative type)* is a type if an object of the type is always in collaboration (association) with another object.

Ex: Reader

A *role type* is a founded and non-rigid type.

Role types are in collaboration and if the object does no longer play the role type, it does not give up identity

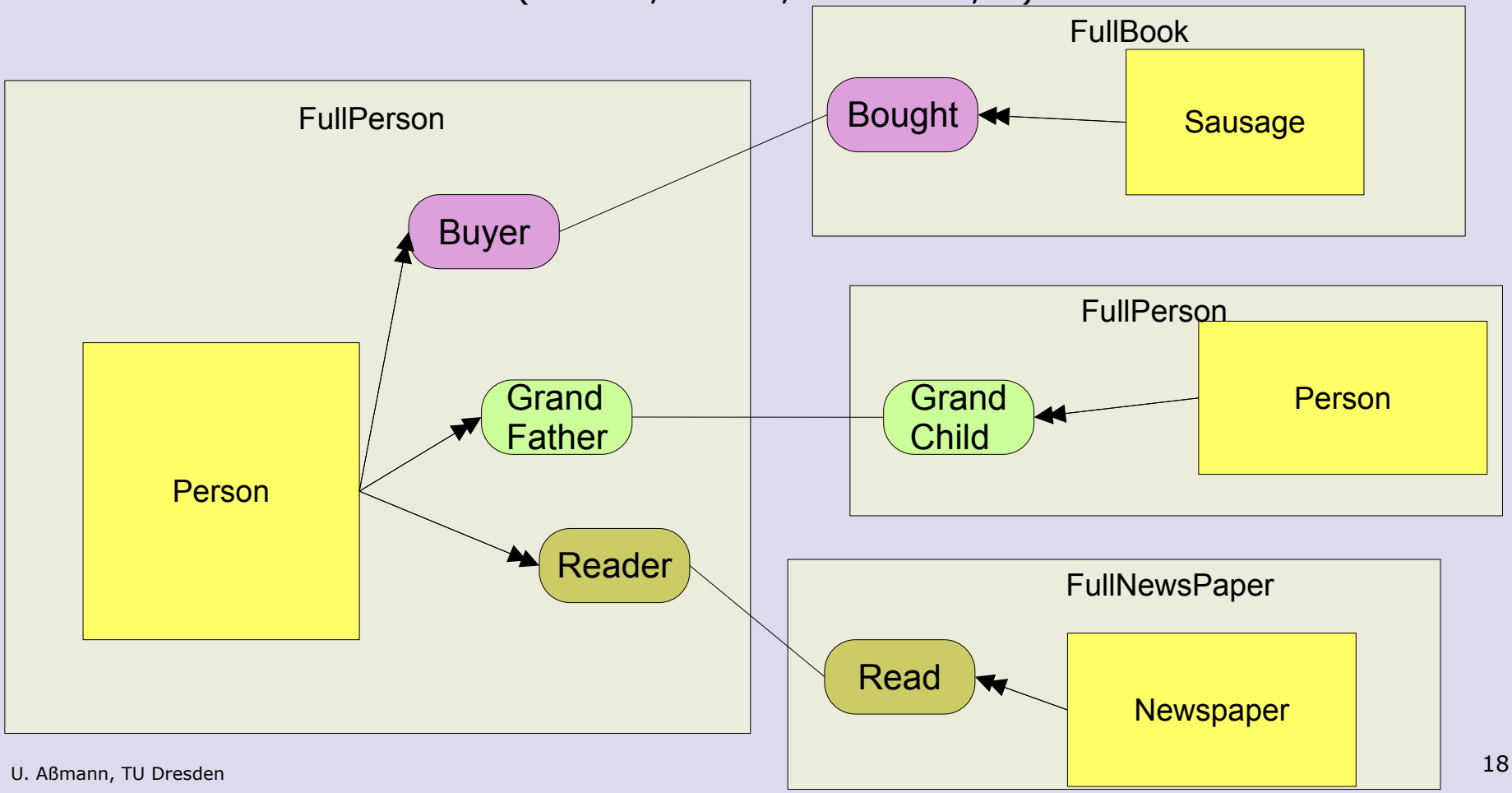
Natural types are non-founded and semantically rigid.

A natural type is *independent* of a relationship
The objects cannot leave it

Steimann Factorization

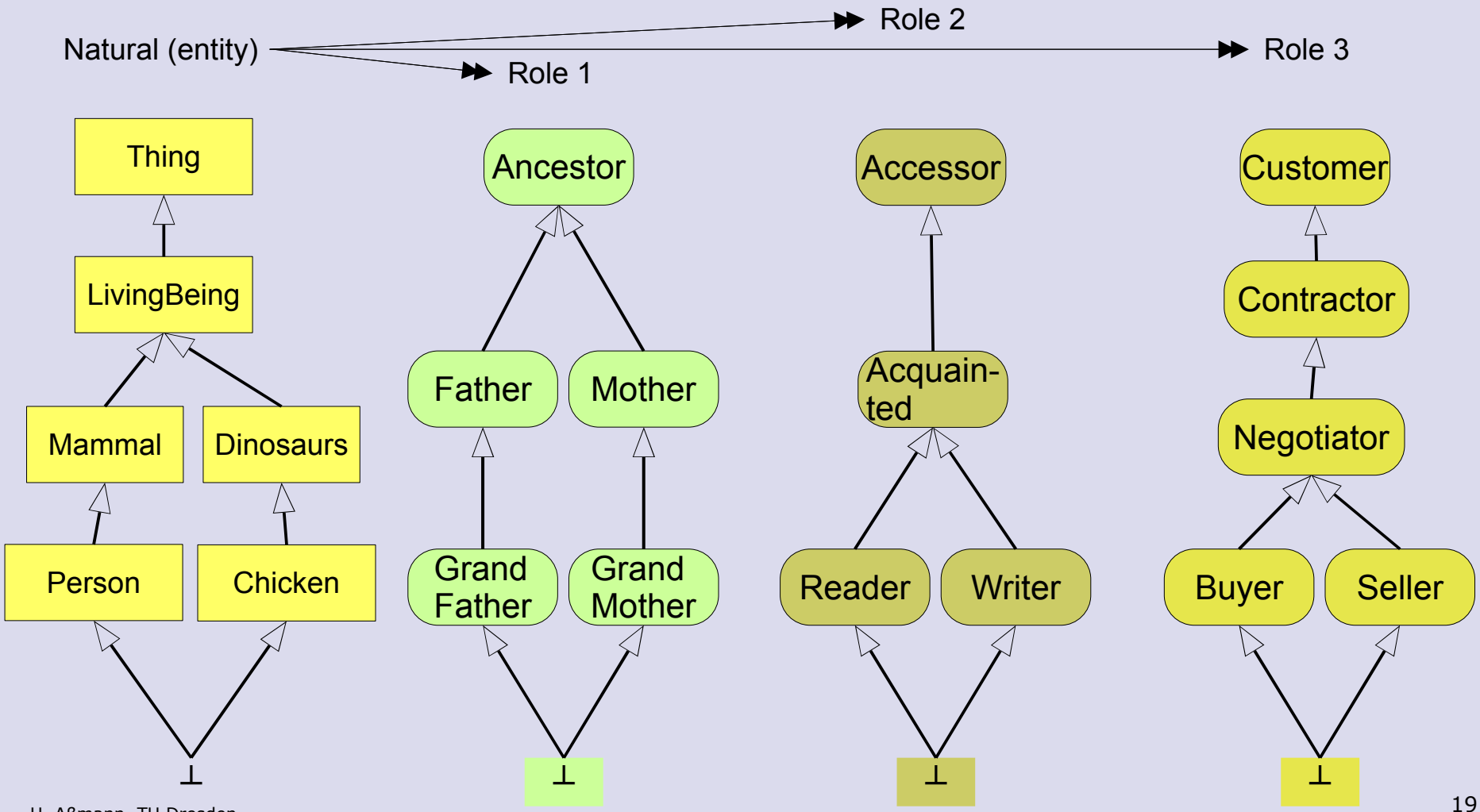
Splitting a full type into its *natural* and *role-type* components

- FullType = Natural x (role-type, role-type, ...)
- FullPerson = Person x (Reader, Father, Customer, ..)



Full Type is from Inheritance Product Lattice

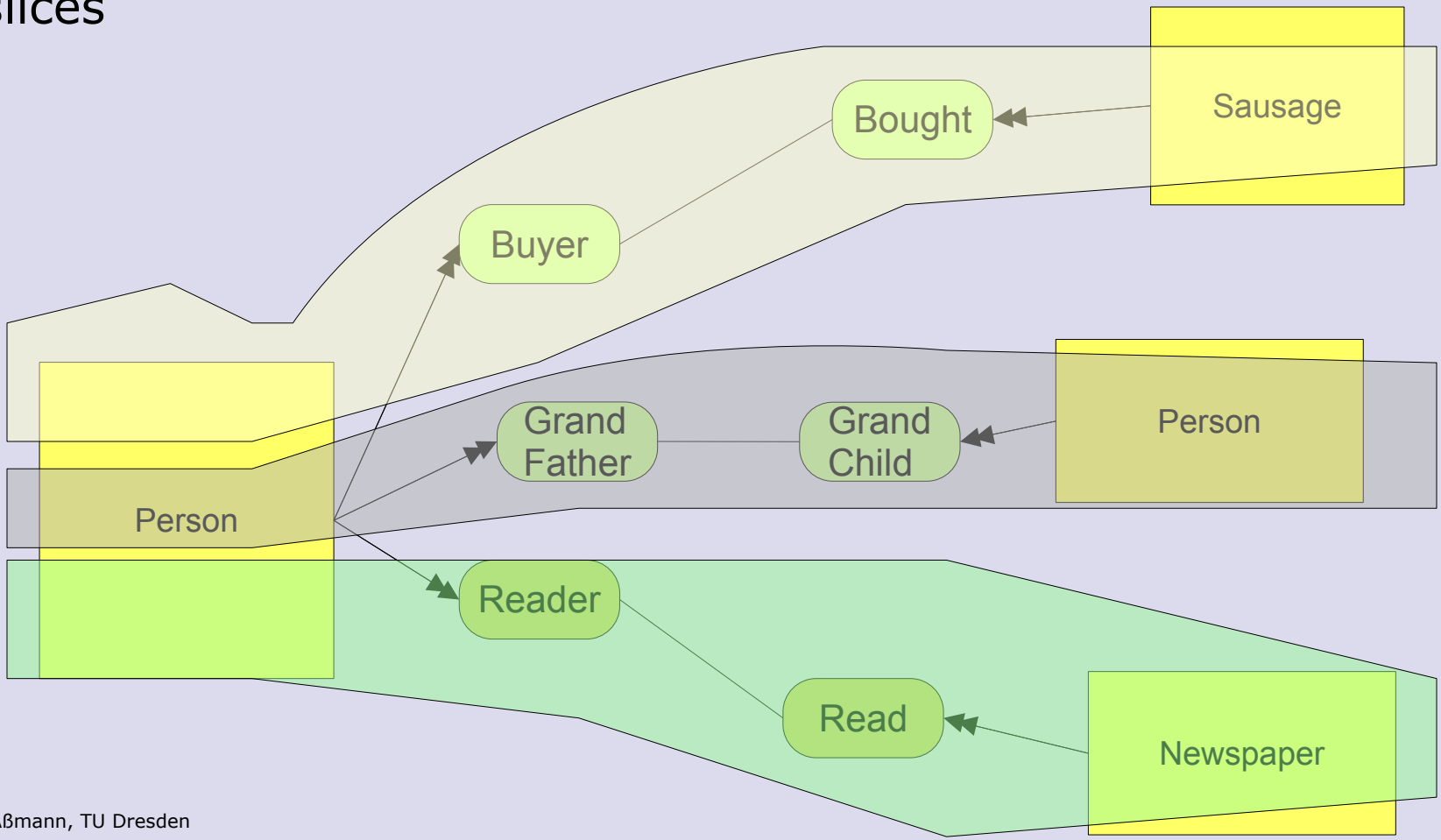
What is a reading buying grandfather person?



Simplified Representation of a Full Type

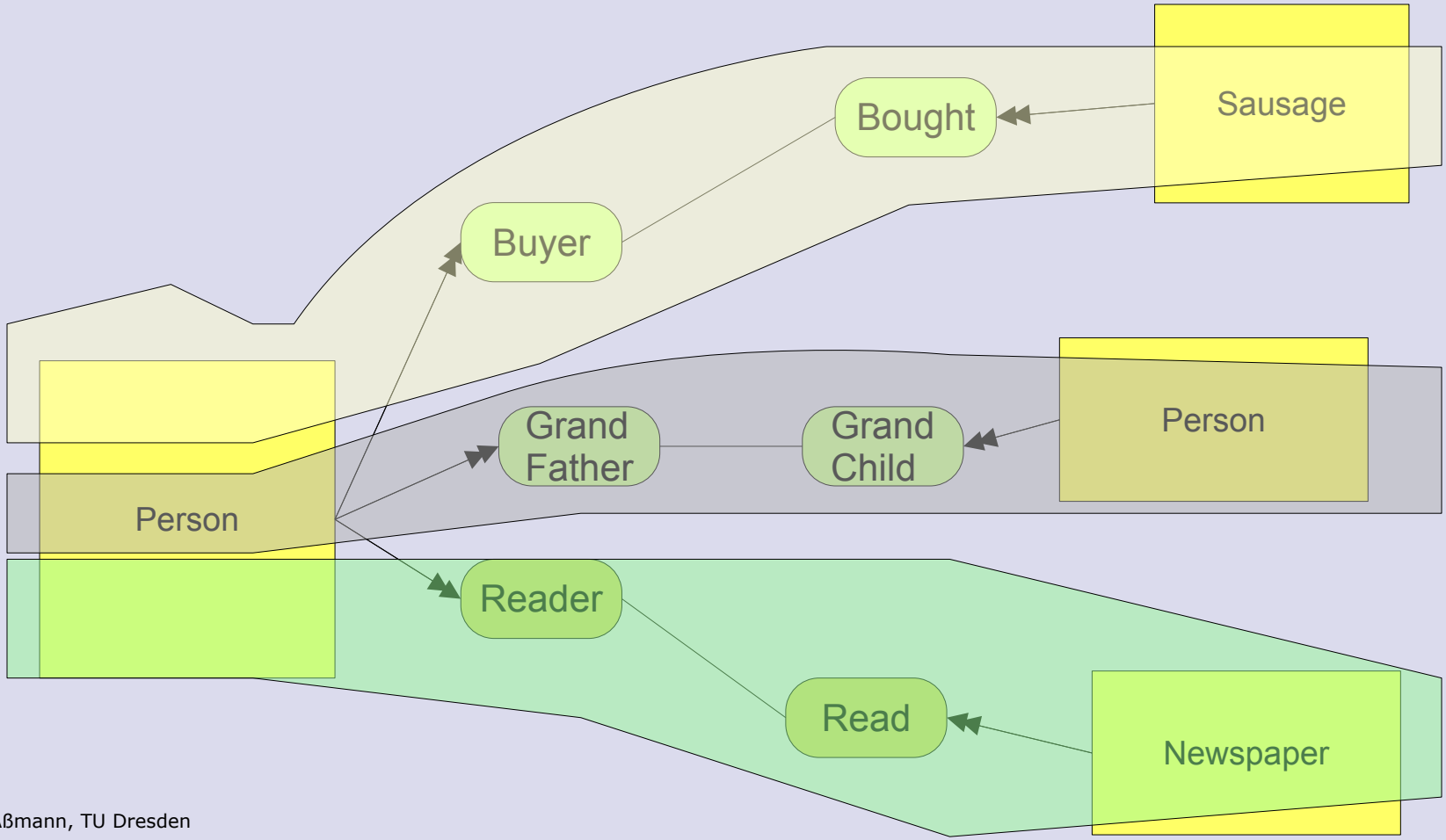
Role models are interprocedural *slices*

Collaboration schemas are schemas (types) for interprocedural slices



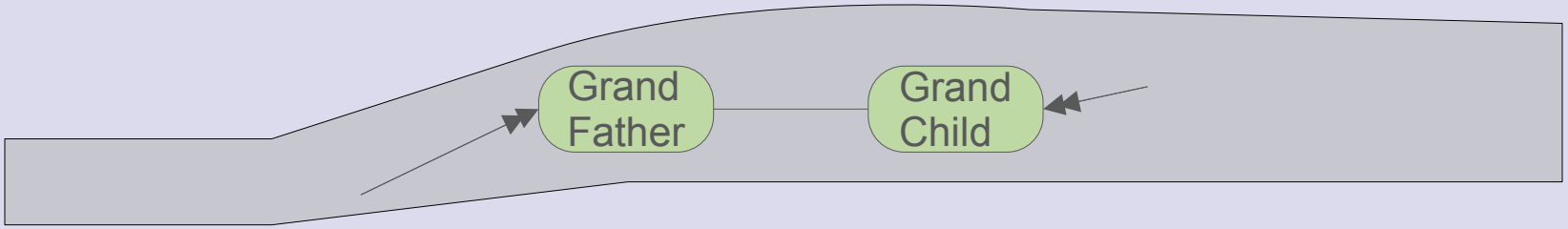
Simplified Extension

Collaboration schemas can be *extended by* new ones

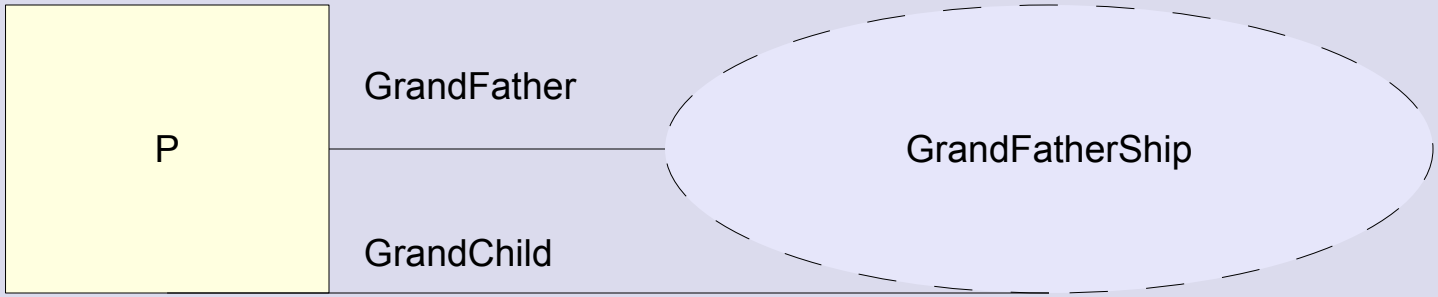


A Collaboration Schema is a Relational Module

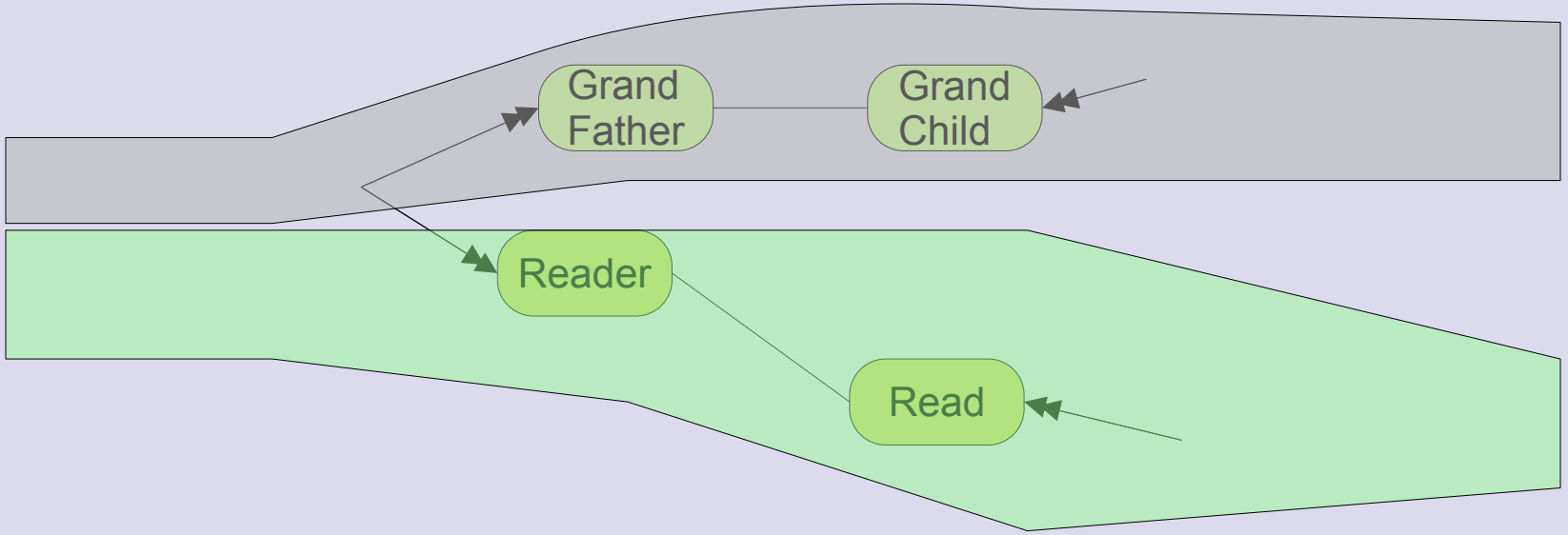
The open ends are the *plays-a* tentacles

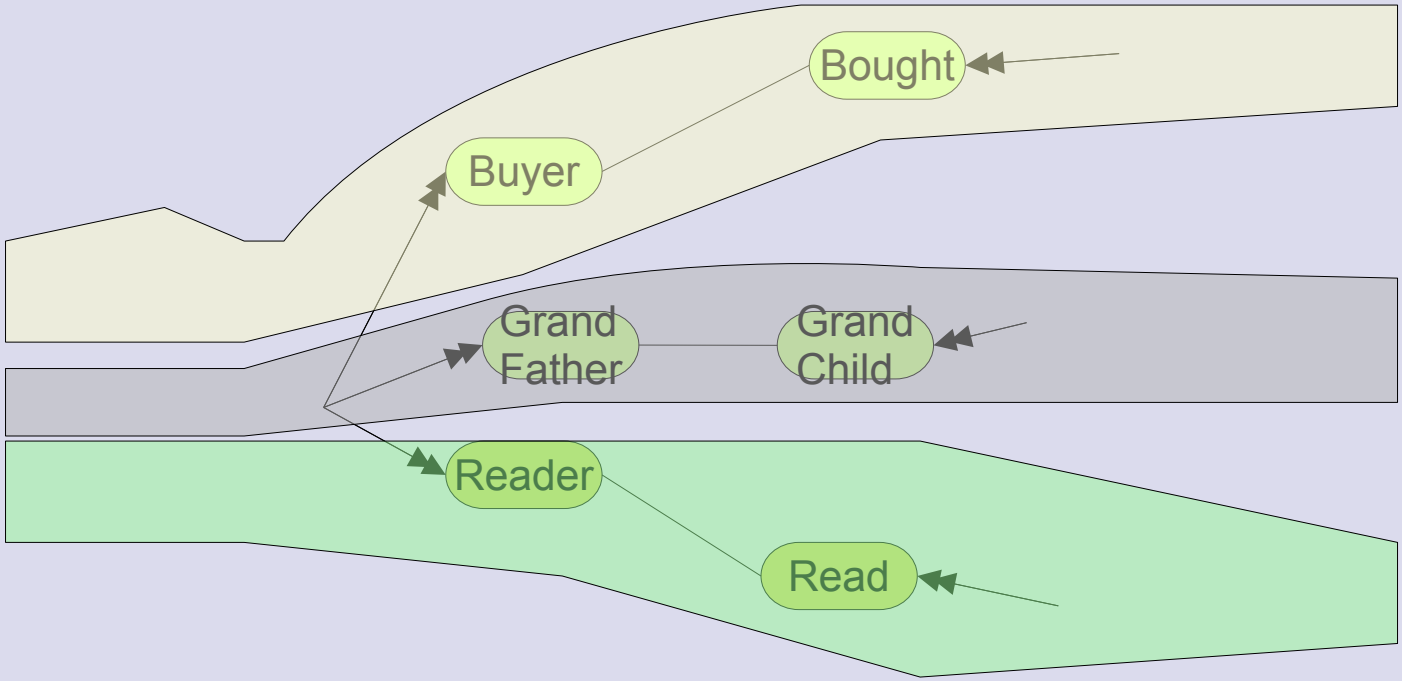


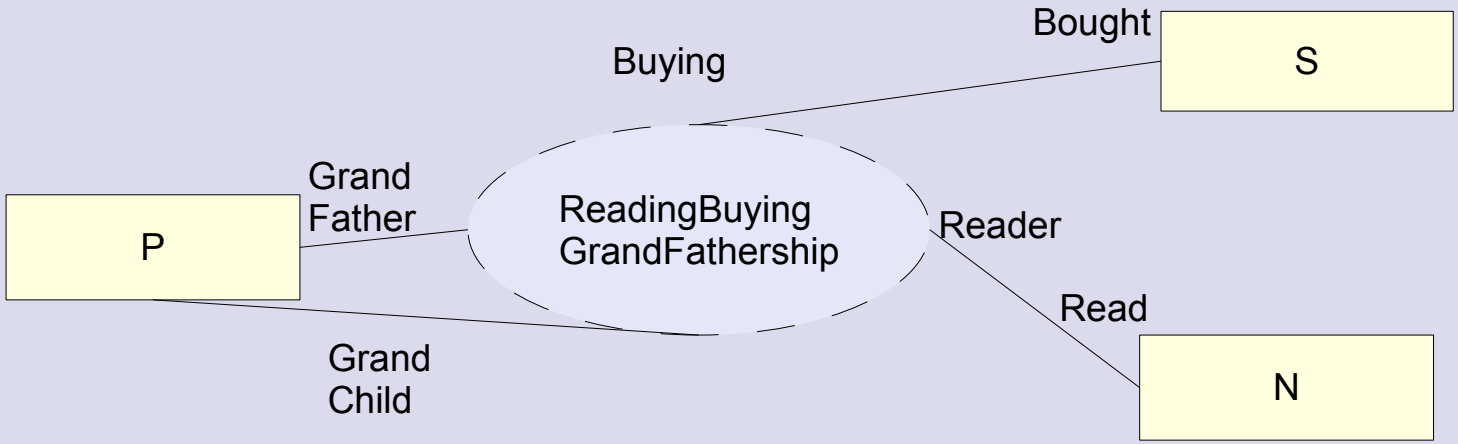
UML Notation with *role-type parameter* P:



Newspaper-Reading GrandpaShip




Bought
Buyer
Grand
Father
Grand
Child
Reader
Read


Bought
Buying
S
Grand
Father
P
ReadingBuying
GrandFathership
Reader
Read
Grand
Child
N

Implementation of Collaboration Schemas

Collaboration schemas are *type functors*

- Functions on types

Direct implementation

- Languages: ObjectTeams, CeasarJ
- Mixin layers
- Role Object Pattern
- Semantic macros
- Generic templates (BETA, Compost)
- Aspects

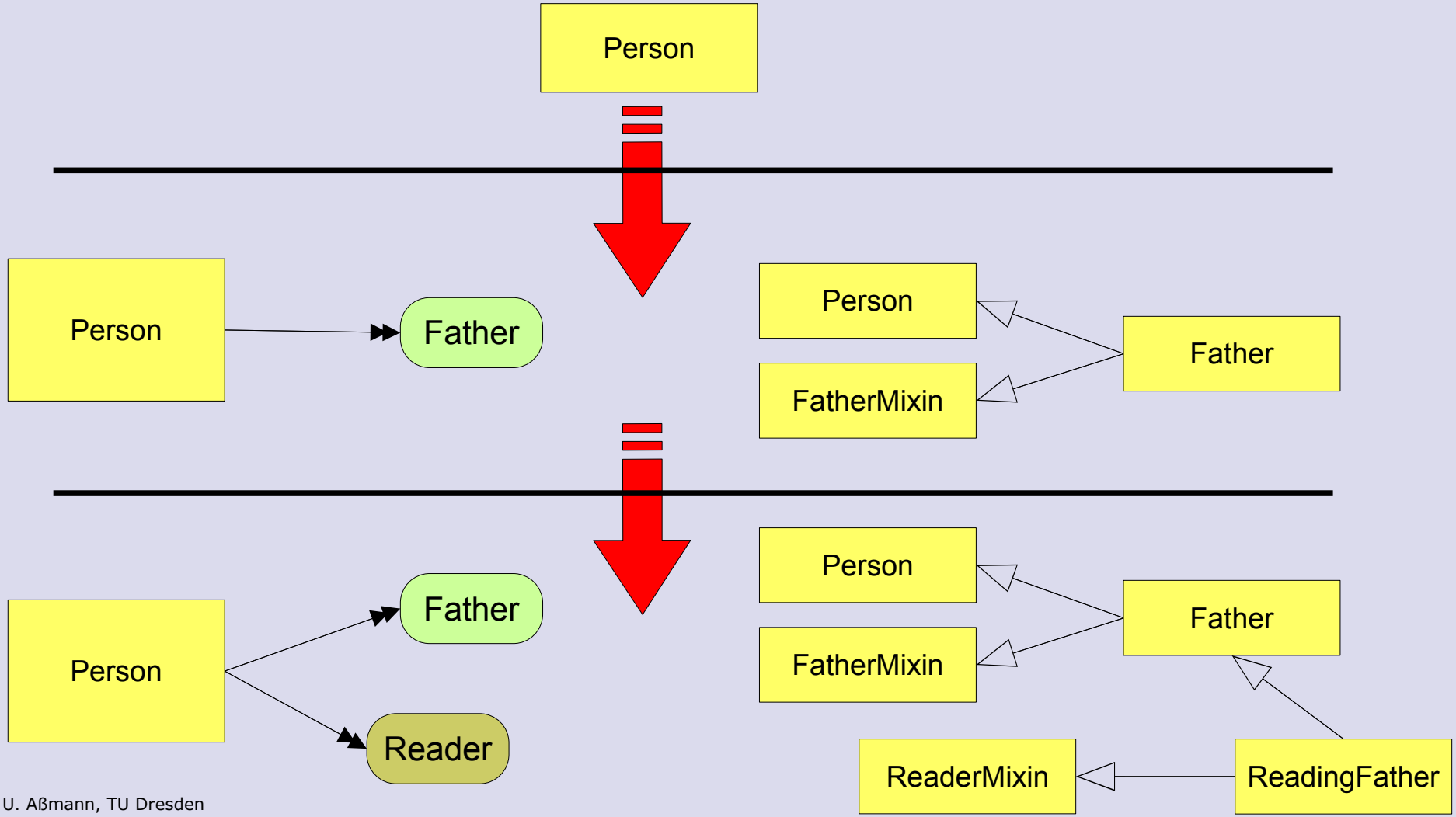
Rewriting to standard languages

- Mapping (MDD process), e.g., with graph rewriting systems

Why Role Extension Retains Identity of a Natural Type

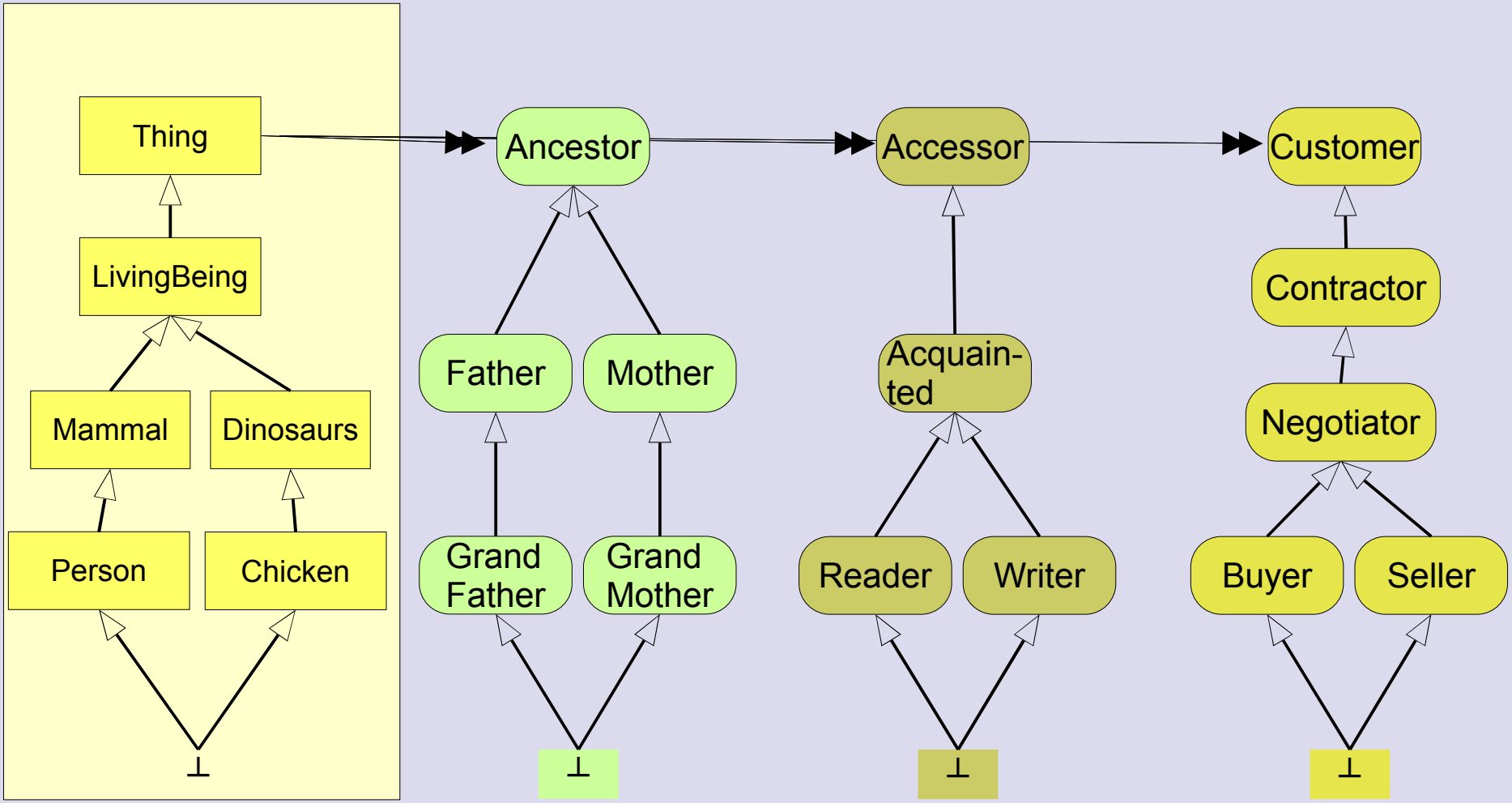
Role Extension Retains Identity

Role types are NON-RIGID



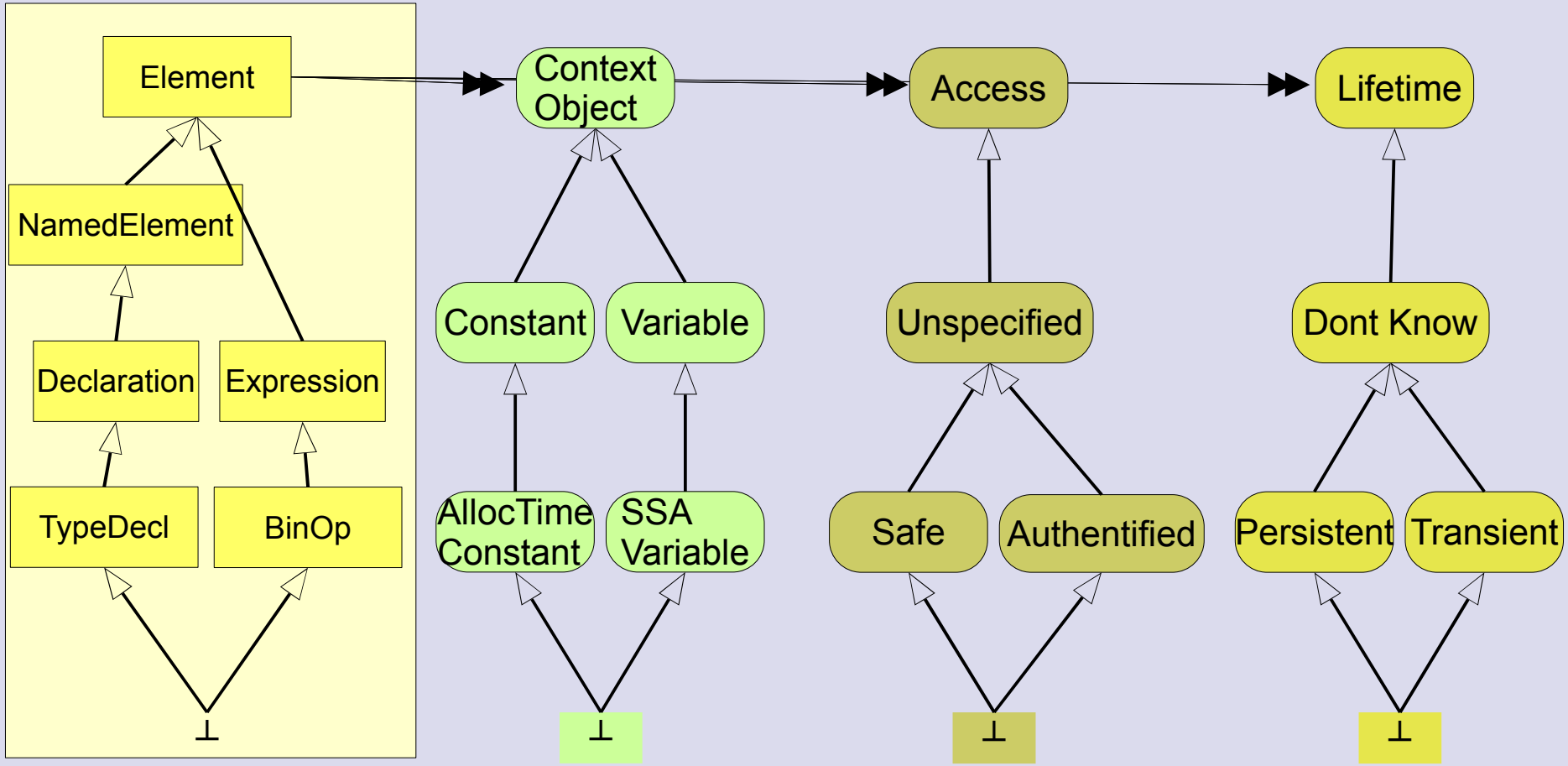
Identity is Fixed to Core Facet of Product Lattice

Role type extensions do not change the name of the full type



Identity is Fixed to Core Facet of Product Lattice

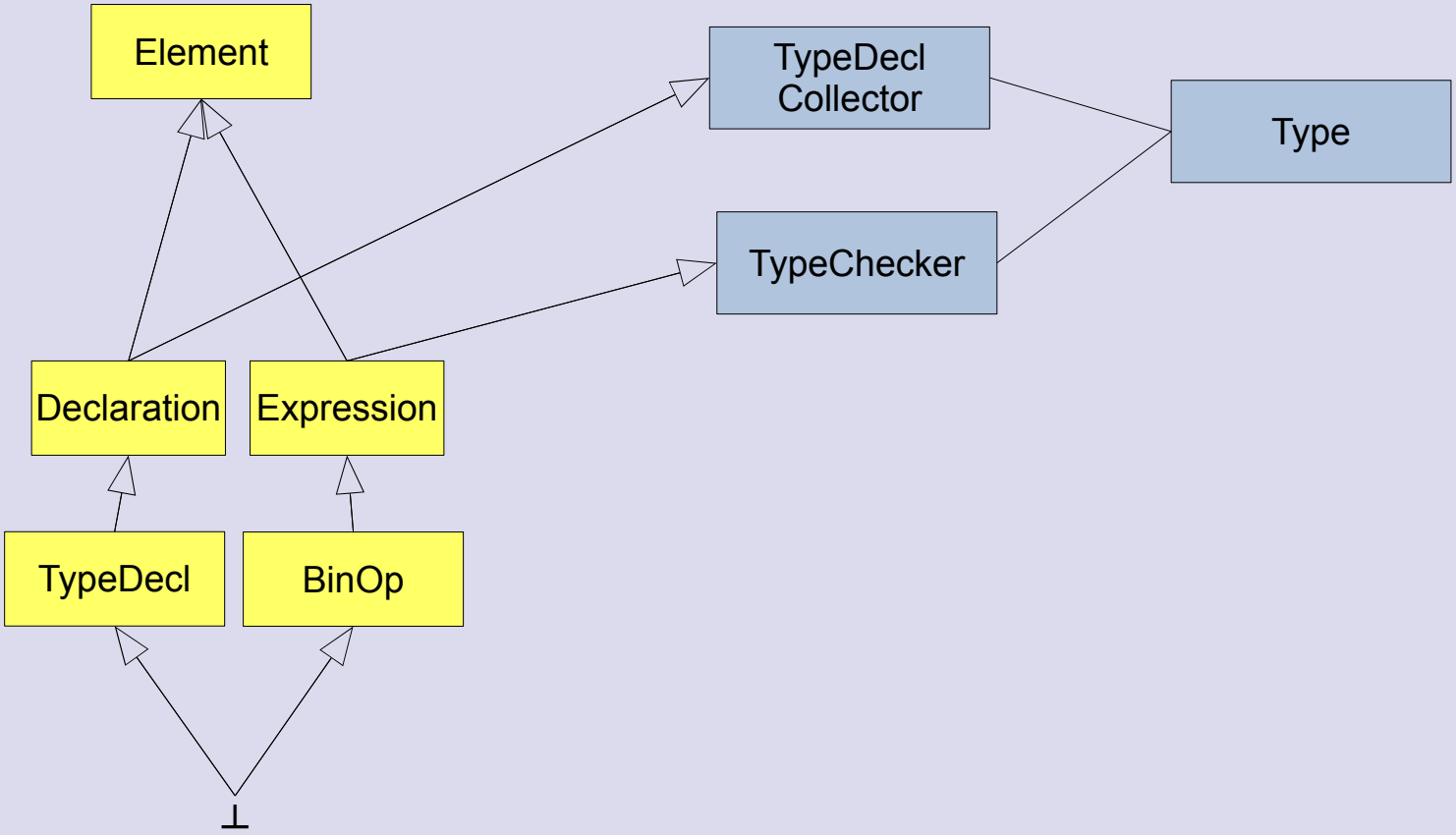
Role type extensions specify the behavior of a *language concept in context*



Problems In Language Composition

Superclass Superimposition

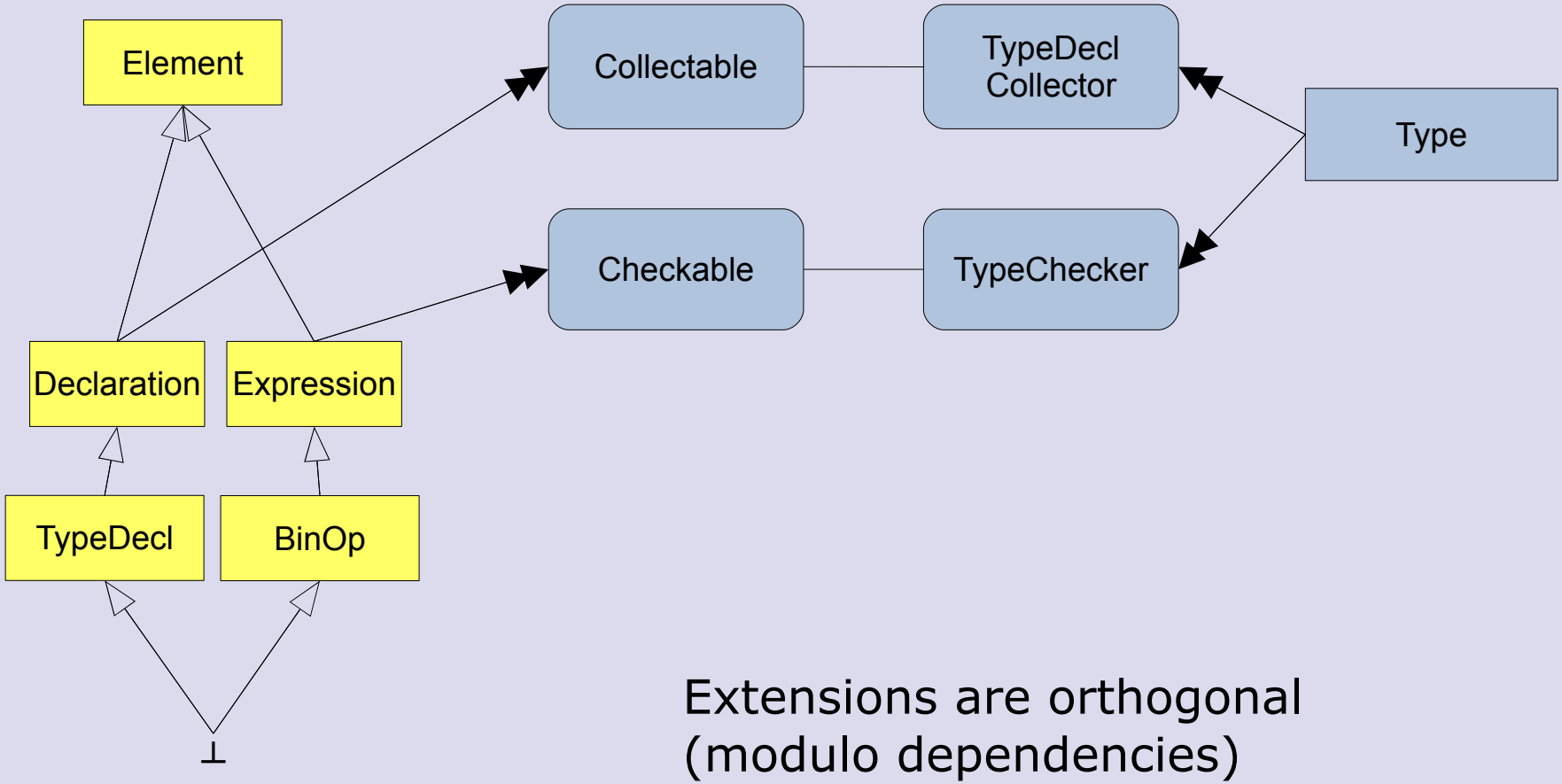
How to add a new interface/role to existing class hierarchy?



Good News: Role Superimposition EASY

Roles are transient, non-rigid, identity-preserving

- Entity inheritance hierarchies are preserved

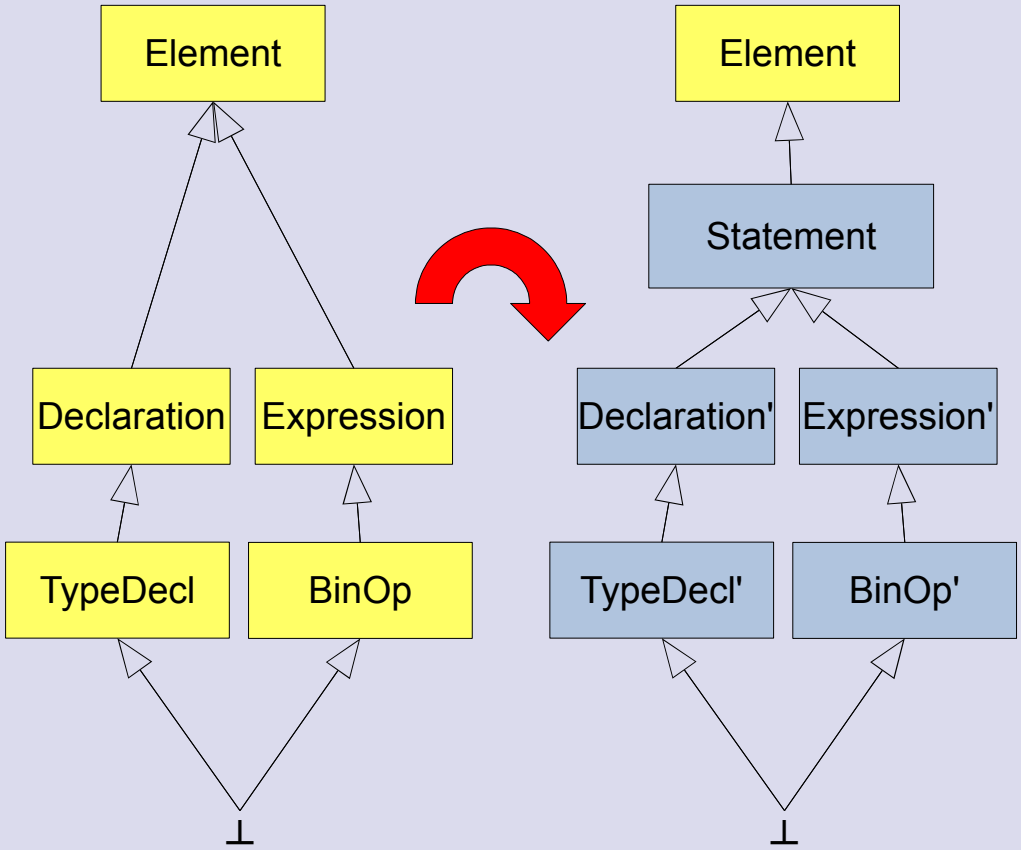


Extensions are orthogonal
(modulo dependencies)

Bad News: Superimposition of Entity Natural Superclasses Stays HARD

Identity of all derived subclasses changes

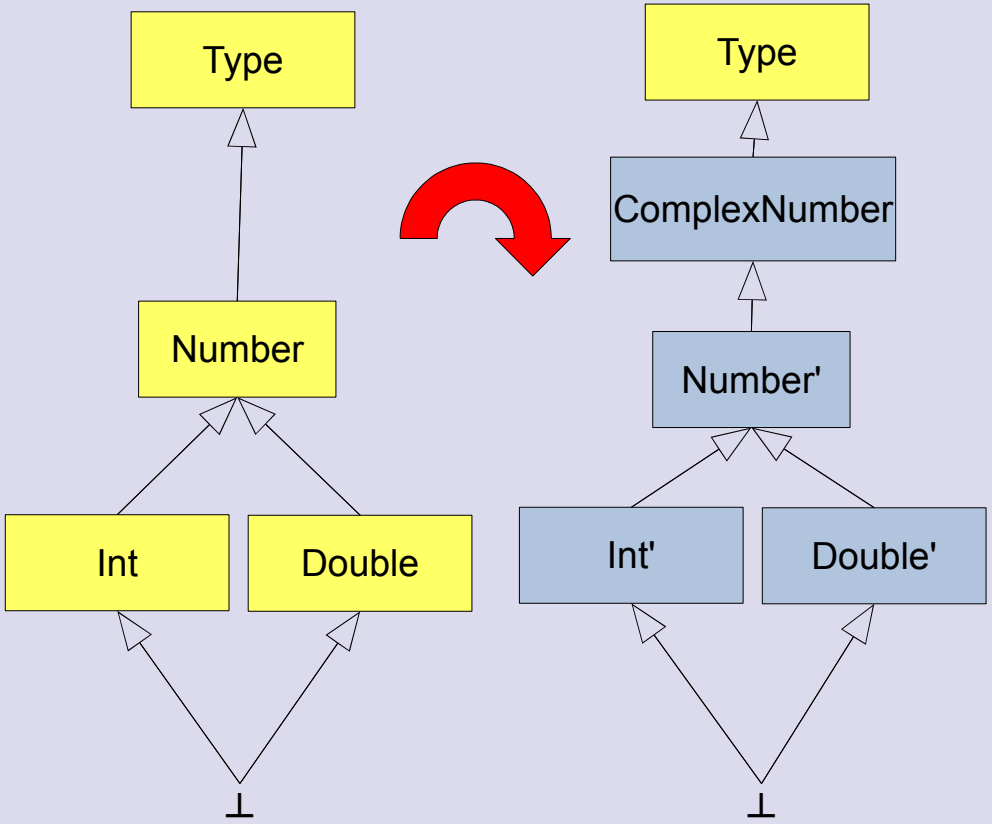
- Declaration --> Declaration' under-a Statement
- Expression --> Expression' under-a Statement



Example: Complex Numbers

Superimposing a new concept ComplexNumber to a type hierarchy is an extension of the entity (natural) concepts of a language

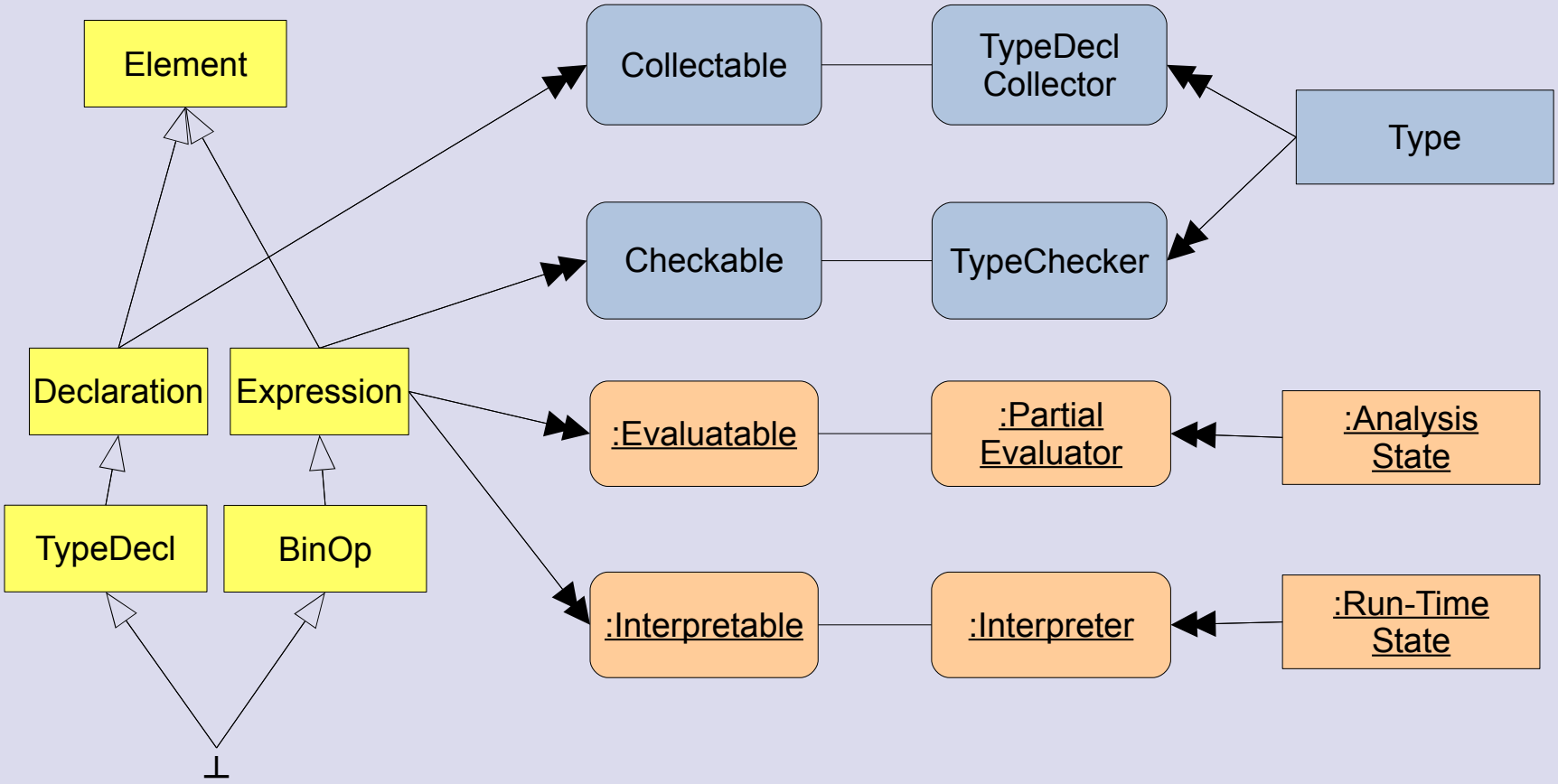
- Due to identity change, type rules for all Numbers have to be changed [van Wyk, JLE application]



Dynamic Semantics can be Composed as Role Models

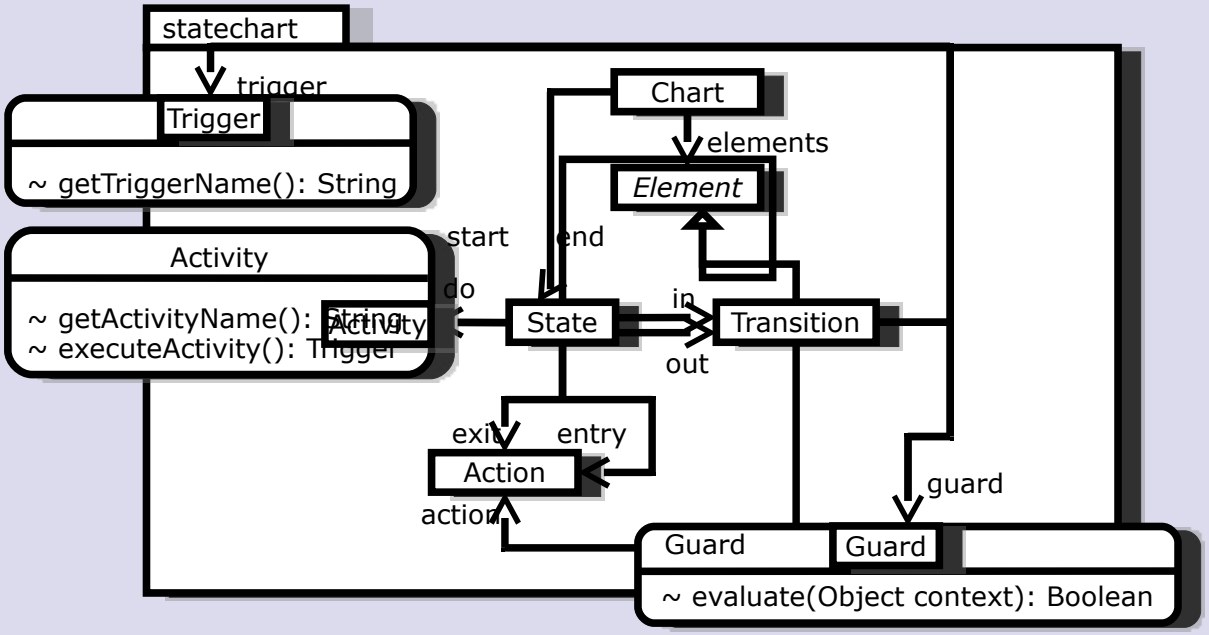
With dynamic composition of *role models*

- instead of static composition of *collaboration schemas*



The LanGems Language Composition System

LanGems: Role-based Language Specification



Abstract Syntax of a language module are specified by a type collaboration

Natural — *Naturals* define rigid types with stable identity which have properties, interrelations to other types and behaviour (semantics, not meant to vary

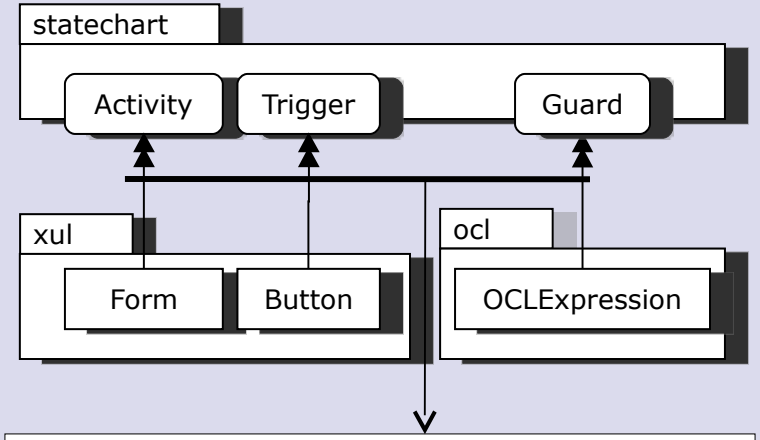
Role — *Roles* define *Variation Points* in a language module

Roles obtain identity and behaviour from a role player outside the module, can also contribute own have properties, interrelations to other types and behaviour (semantics)

~ roleOp() — *RoleOperations* define semantic binding point for role players, a role-playing contract

Module Composition Language

- Types contribute the composition interface of language modules
  - Role Types: required interface
  - Natural Types: provided interface
- Language Composition is described by superimposition of the collaborations of several modules where *RoleBinding* (↠) connects role player and role
- Binding of RoleOperations in the context of a role player (*RoleOperationBindings)* contributes structual and semantic adaptation of the role player w.r.t the role contract

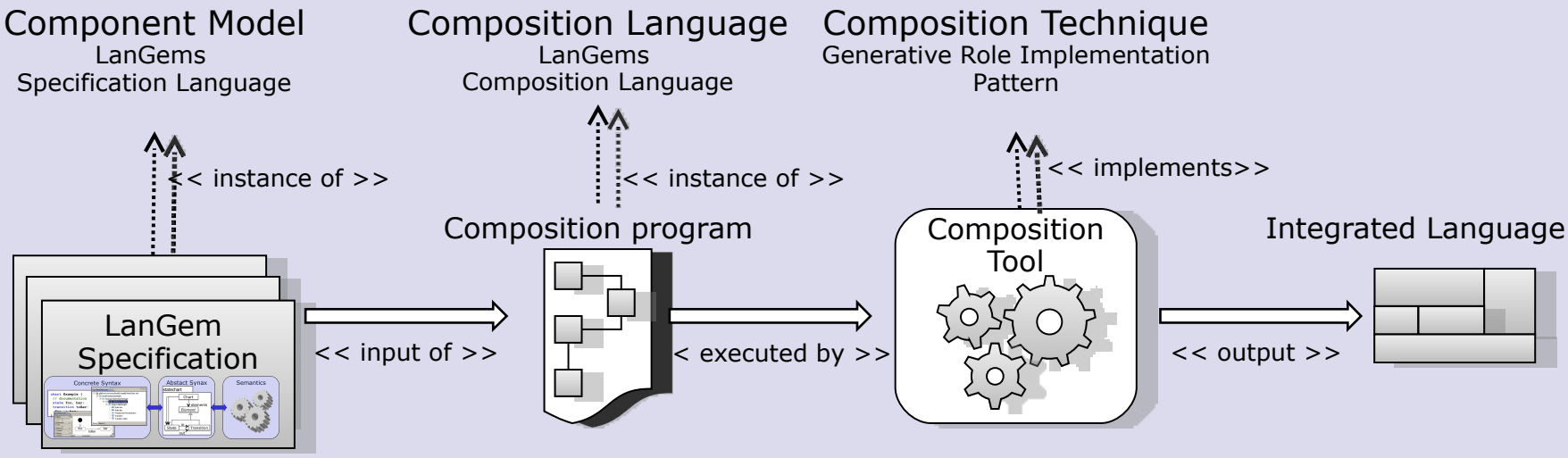



```
Form plays Activity {
   getActivityName(): player.getTitle();
   executeActivity (): player.open();
}

Button plays Trigger {
  getTriggerName(): player.getText();
}

OCLExpression plays Guard {
  evaluate(Object context) : if (player.type = boolean)
                             then player.interpret(context)
                             else false;
}
```

LanGems Composition System



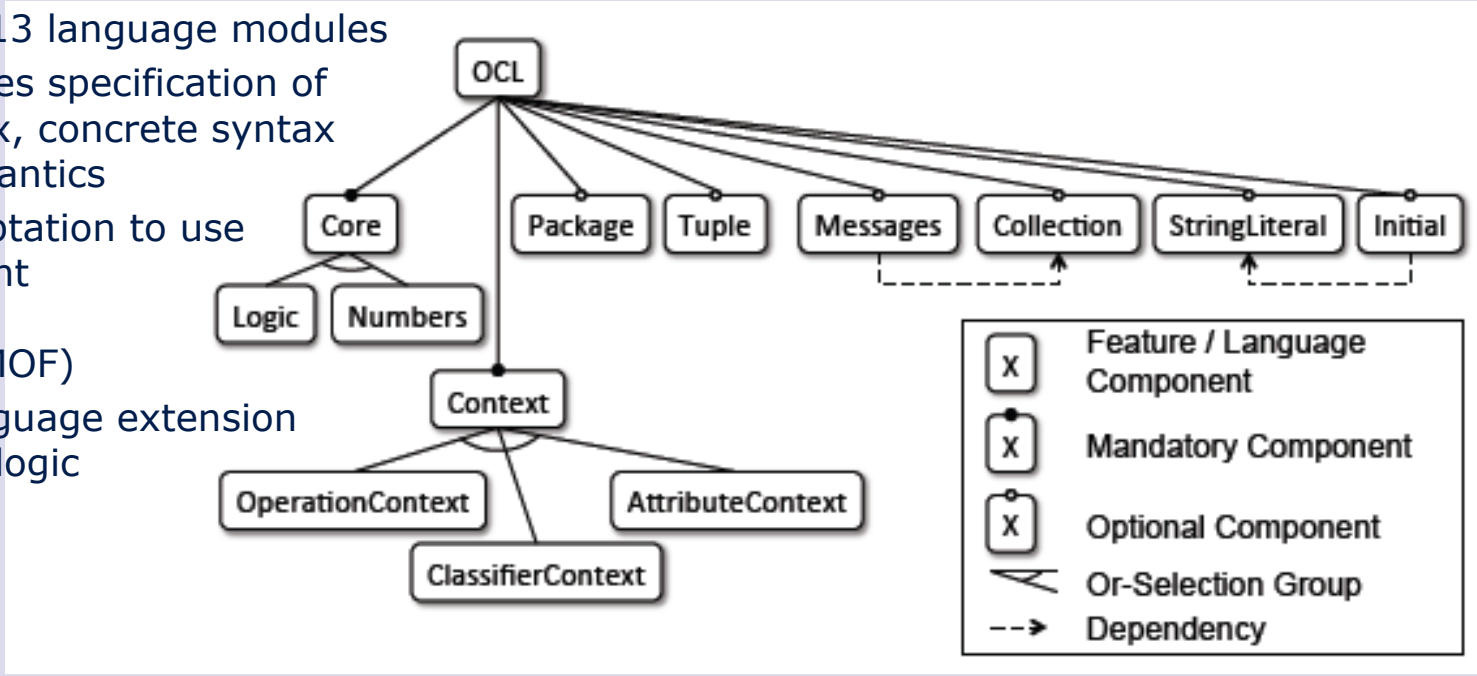
Case Study: Modularisation of OCL

OCL

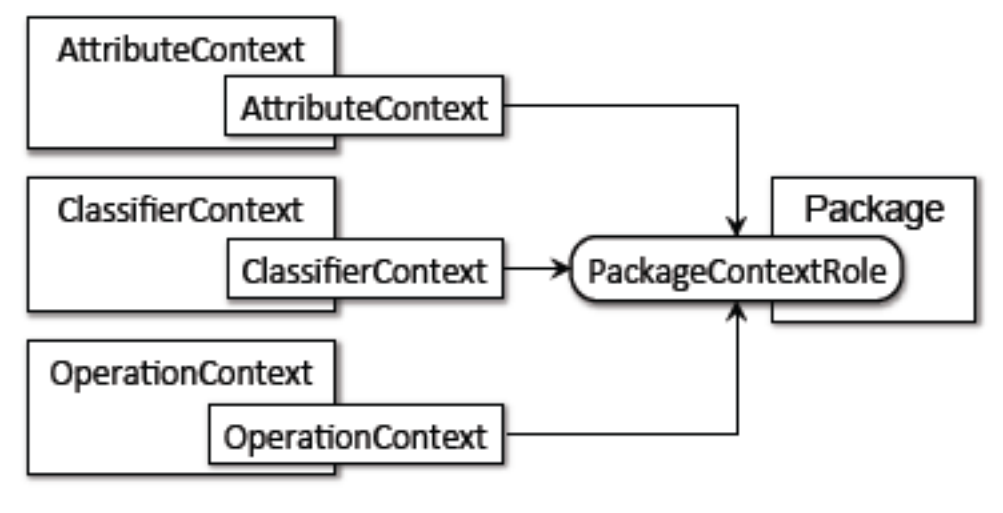
- Complex language
- Applied at different abstraction layers and environments
- Several proposals for extension of OCL

Activities

- Separation of 13 language modules
- Each contributes specification of abstract syntax, concrete syntax and static semantics
- Language adaptation to use OCL on different metamodels (Ecore, UML, MOF)
- Exemplary language extension with temporal logic



Evaluation



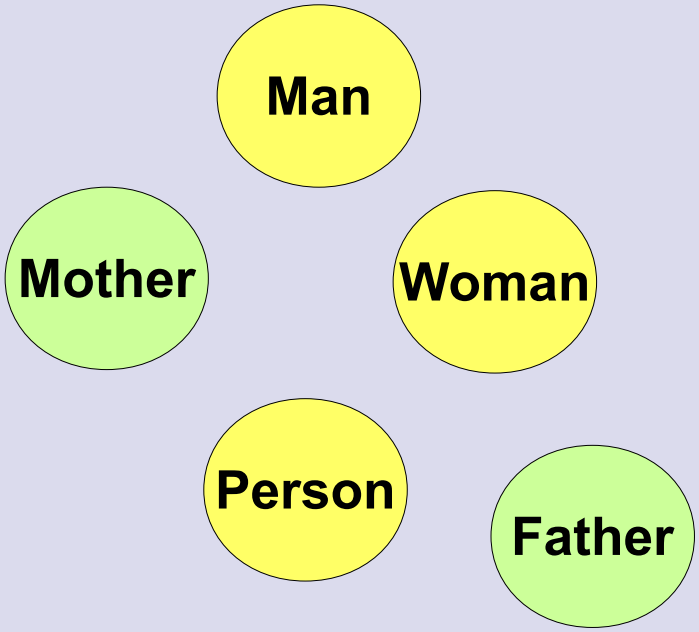
Experienced Benefits

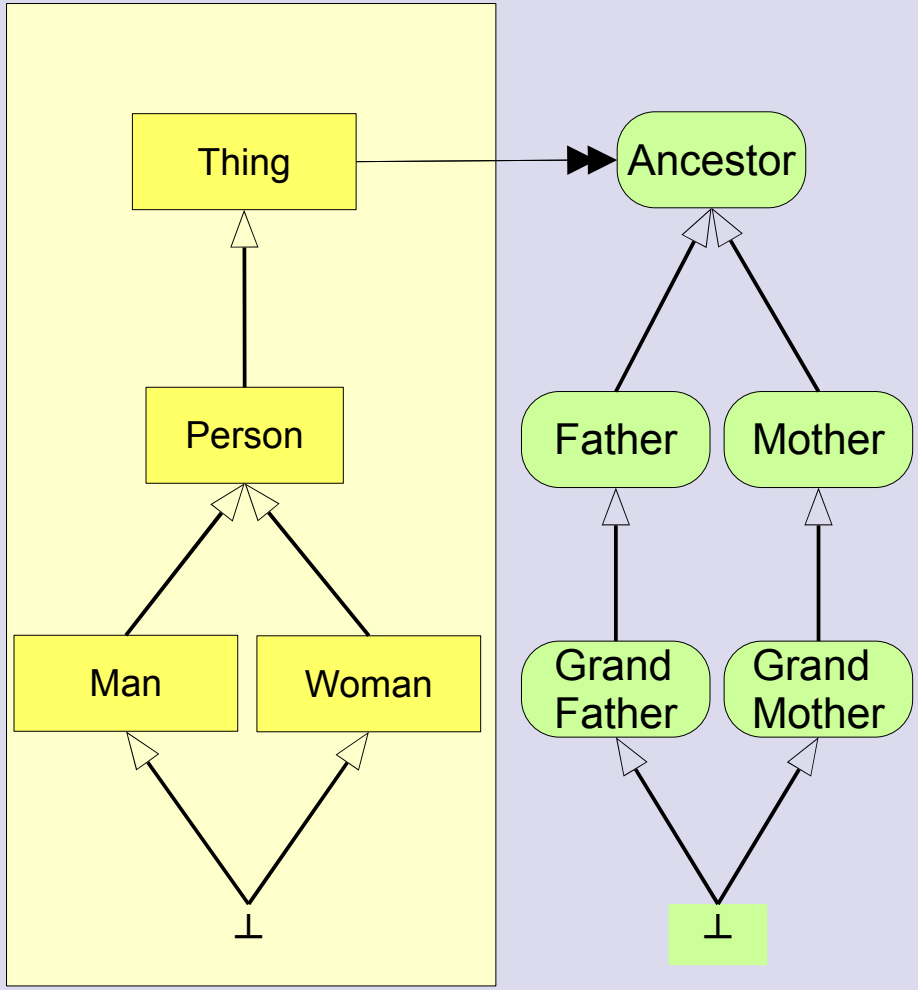
- Self-contained comprehensible modules
  - Independent Development and Maintenance
  - Explicit language component interfaces decouple language modules
  - Adaptation of OCL by variation on language modules
  - Extension of OCL by adding language modules
- Role-based modularisation and composition supports for concrete syntax and language semantics
  - Composition did not invalidate module syntax and semantics
  - Composition provides means for semantic (and structural) adaptation

Problems & Open Issues

- Operator priorities needs to be considered during composition
- Context-free parsing required adjustment of token definitions among modules
- Dynamic Semantics not implemented yet

Solution to the Little Riddles..





Summary

Collaboration schemas (role-type models) offer

- *relational modules* for model and language composition
- *extensibility* for M1 models, classes, programs can easily be lifted to M2 metamodels (language specifications)

The Steimann Factorization (natural – roles) allows for

- simple extension static semantic specifications, because role extension does not change identity of objects or concepts
- simple extension of interpreters (dynamic semantics)

Entity extension - of *natural* language concepts - stays hard

Role-based language composition system LanGems

- allows to define encapsulated language components
- provides contractually composition interfaces between language components, and
- allows to compose and reuse language components individually

End

TUD ST group	http://st.inf.tu-dresden.de
Reuseware toolset	http://www.reuseware.org
ModelPlex project	http://www.modelplex.org
MOST project	http://www.most-project.eu

Other Literature

U. Aßmann. Invasive Software Composition. Springer-Verlag, Feb. 2003.

G. Bracha. The Programming Language Jigsaw: Mixins, Modularity and Multiple Inheritance. Ph.D. thesis, Dept. of Computer Science, University of Utah, Mar. 1992.

G. Bracha and W. Cook. Mixin-based inheritance. In N. Meyrowitz, ed., OOPSLA/ECOOP '90, number 25(10) in ACM SIGPLAN Notices, pages 303 311. ACM Press, New York, Oct. 1990.

G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. ECOOP 97, volume 1241 of LNCS, pp 220 242. Springer, 1997.

S. Krishnamurthi, M. Felleisen, and B. F. Duba. From Macros to Reusable Generative Programming. In U. W. Eisenecker and K. Czarnecki, ed., GCSE, LNCS 1799, Springer 1999.

P. Tarr, H. Ossher, W. Harrison, and S. Sutton. N degrees of separation: Multi-dimensional separation of concerns. In Proceedings of ICSE'99, pp 107-119, 1999

Literature on Roles

T. Reenskaug, P. Wold, O. A. Lehne. Working with objects. Manning. The OOram Method. http://heim.ifi.uio.no/~trygver/documents/book11d.pdf

H. Allert, P. Dolog, W. Nejdl, W. Siberski, F. Steimann. Role-Oriented Models for Hypermedia Construction – Conc*eptual Modelling for the Semantic Web.* citeseer.org.

N. Guarino, M. Carrara, and P. Giaretta. An ontology of meta-level categories. In Proceedings of the Fourth International Conference on Knowledge Representation and Reasoning, pages 270–280. Morgan Kaufmann, San Mateo, 1994.

F. Steimann. On the representation of roles in object-oriented and conceptual modelling. Data and Knowledge Engineering. 2000.

T. Reenskaug, P. Wold, O. A. Lehne. Working with objects. Manning. http://heim.ifi.uio.no/~trygver/documents/book11d.pdf

D. Riehle, T. Gross. Role Model Based Framework Design and Integration. OOPSLA 1998.

U. Aßmann, J. Henriksson, I. Savga, J. Johannes: Composition of Ontologies and Rule Sets. REASONING WEB Summer School, LNCS 4126.

REWERSE
reasoning on the web

The End

REWERSE Working Group I3 (*Composition and Typing*) **http://www.rewerse.net**

TUD ST group	**http://st.inf.tu-dresden.de**
Reuseware toolset	**http://www.reuseware.org**
ModelPlex project	**http://www.modelplex.org**
MOST project	**http://www.most-project.eu**