University of Applied Sciences and Arts Northwestern Switzerland
School of Business

KEBI Knowledge Engineering & Business Intelligence

MSc Business Information Systems

University of Applied Sciences and Arts Northwestern Switzerland
School of Business

Combining Object-centered Representation and Logic Programming

MSc Business Information Systems

ObjectLogic

- ObjectLogic is a deductive, object oriented database language which combines
  - the declarative semantics and expressiveness of logic programming
  - with the rich data modeling capabilities supported by the object oriented data model.
- ObjectLogic is a kind of successor of F-Logic, a kind of successor of PROLOG

ObjectLogic: Schema-level Statements

- Signature F-atoms define methods of classes
- Methods correspond to roles in Description Logic

Subclass Relation

```
Car::Vehicle.
Bike::Vehicle.
```

Every car is a vehicle
Every bike is a vehicle

Signature statements (methods, cardinalities and ranges)

```
Person[name {1:*} *=>  xsd#string].
Person[friend {0:*} *=>  Person].

Vehicle[owner {1:*} *=> Person].
```

A person has at least one name, which is a string

A vehicle has a owner which is a person

ObjectLogic: Schema-level Statements

- Several signature-F-atoms may be combined in an F-molecule

```
Vehicle[owner {1:*} *=>  {Person, Adult}].
```

This is equivalent to the conjunction of the two lines:

```
Vehicle[owner {1:*} *=> Person].
Vehicle[owner {1:*} *=> Adult].
```

- In the following signature-F-atom the method **owner** has a parameter object of the datatype **integer**. It should denote the year of ownership.

```
Vehicle[owner(xsd#integer) {1:1} *=>  Person].
```

ObjectLogic: Instance-level Statements

- The application of a method on an object is expressed by data-F-atoms which consist of a host object, a method and a result object
- Relating instances to classes

```
peter:Man.
paul:Person.
car74:Car.
```

peter is of class man
paulis of class person
car74is of class car

- Method invocation: assigning values to methods

```
paul[friend -> mary].
peter[friend ->{paul, mary}].
car74[owner(2007) -> paul].
car74[owner(2008) -> peter].
```

paul has a friend who is mary
peter has friends paul and mary

the owner of car74 are paul in year 2007 and peter in year 20089

- Combining class associations and method invocations

```
peter:Man[friend-> mary:Woman].
```

Peter is of class man and has a friend who is mary, who is a woman.

Abbreviations

- Instead of giving several individual atoms, information about an object can be collected in F-molecules, which combine multiple F-atom statements in a concise way
- Assignments about a single object can combined into one expression

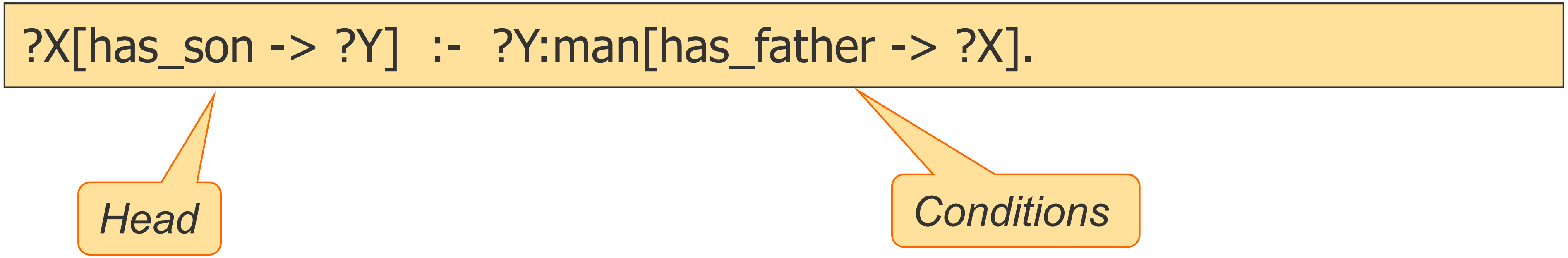
```
jacob:man[has_father -> isaac;
           has_son -> {joseph:man,
                       benjamin:man[has_mother -> rahel]}].
```

is equivalent to:

```
jacob:man.
joseph:man.
benjamin:man
jacob[has_father -> isaac].
jacob[has_son -> joseph].
jacob[has_son -> benjamin].
benjamin[has_mother -> rahel].
```

ObjectLogic: Rules

- Rules consist of a head and a body – as in logic programming
  - Variables start with a question mark «?»
- Example:



```
?X[has_son -> ?Y]  :-  ?Y:man[has_father -> ?X].
```

Head

Conditions

„For all X and Y : X has a son Y, if Y is a man and has the father X. "

ObjectLogic: Rules

```
?X[friend->?Y]                :- ?Y:Person[friend->?X:Person].

?X[admissibleDriver->?Y] :- ?X:Vehicle[owner->?Y:Person].

?X[admissibleDriver->?Z] :- ?X:Vehicle[owner->?Y]  AND
                                     ?Y:Person[friend->?Z:Person].
```

- Rule with Negation

```
?X[prohibitedDriver->?Y]   :-    ?X:Car AND
                                    ?Y:Person AND
                                    NOT ?X[admissibleDriver -> ?Y].
```

Queries

- Queries are rules without a head
- Example:

```
?- car74[admissibleDriver -> ?Y].
```

Complex Queries

- More complex queries can be formulated that also contain arbitrary first-order formulas in the (rule) body
- The following query computes the maximum value **?X** for which **p(?X)** holds. The rule body expresses that all **?Y** for which **p(?Y)** (also) holds must be less or equal to the searched **?X**.

```
p(1).
p(2).
p(3).
?- p(?X) AND (FORALL ?Y (p(?Y) --> ?Y <= ?X)).
```

- The result is

  ?X = 3.0

Namespaces

- In OntoStudio each object name is a URI (uniform resource indicator)
- A URI looks like a legal internet address. It starts consists of a protocol identifier followed (e.g. http: oder ftp:) followed by an address, e.g.
  - http://www.ietf.org/rfc/rfc791.txt
  - http://www.w3.org/People/Berners-Lee
- The „#“ refers to the default namespace, i.e. the namespace of the current knowledge base
- An ObjectLogic file can contain namespace declarations that associate namespace URIs with aliases

```
:- prefix cars="http://www.cars-r-us.tv/".
:- prefix finance="http://www.financeWorld.tv/".
:- prefix xsd="http://www.w3.org/2001/XMLSchema#".
:- default prefix ="http://www.myDomain.tv/private#".
```

Namespace Expressions

- Every concept, method, object, predicate and function may be qualified by a namespace. To separate the namespace from the name the "#"-sign is used

```
cars#Person[   cars#name {1:*} *=>  xsd#string,
               cars#age {1:1} *=>  xsd#integer,
               cars#drivingLicenseId {1:1} *=>  xsd#string].

finance#Bank[  finance#customer {1:*} *=>  finance#Person,
               finance#location {1:*} *=>  finance#City].
```

- During parsing of the ObjectLogic program the aliases are resolved
  - finance#Person   *stands for*   <http://www.financeWorld.tv/Person>
  - cars#Person   *stands fo*   <http://www.cars-r-us.tv/Person>

ObjectLogic vs Logic Programming

- Class names correspond to 1-ary predicates
  - person(X).
  - person(abraham).
- Methods correspond to binary predicates
  - has_father(X,Y).
  - has_father(isaac,abraham).
- Subclass relations correspond to simple rules
  - person(X) :- man(X).
  - person(X) :- woman(X).

ObjectLogic vs. Logic Programming

	ObjectLogic	*Logic Programming*
Subclass definition	man::person	person(X) :- man(X).
Signature statement	person[has_father *=> man]	---
Instances	abraham:man isaac:man sarah:woman	man(abraham). man(isaac). woman(sarah).
Method invocation	isaac[has_father->abraham]. isaac[has_mother->sarah]. abraham[has_son->>isaac].	has_father(isaac,abraham). has_mother(isaac,sarah). has_son(abraham,isaac).

ObjectLogic vs. Logic Programming

- Rules correspond to Horn clause rules

```
?X[has_son -> ?Y]  :- ?Y:man[has_father -> ?X].
```

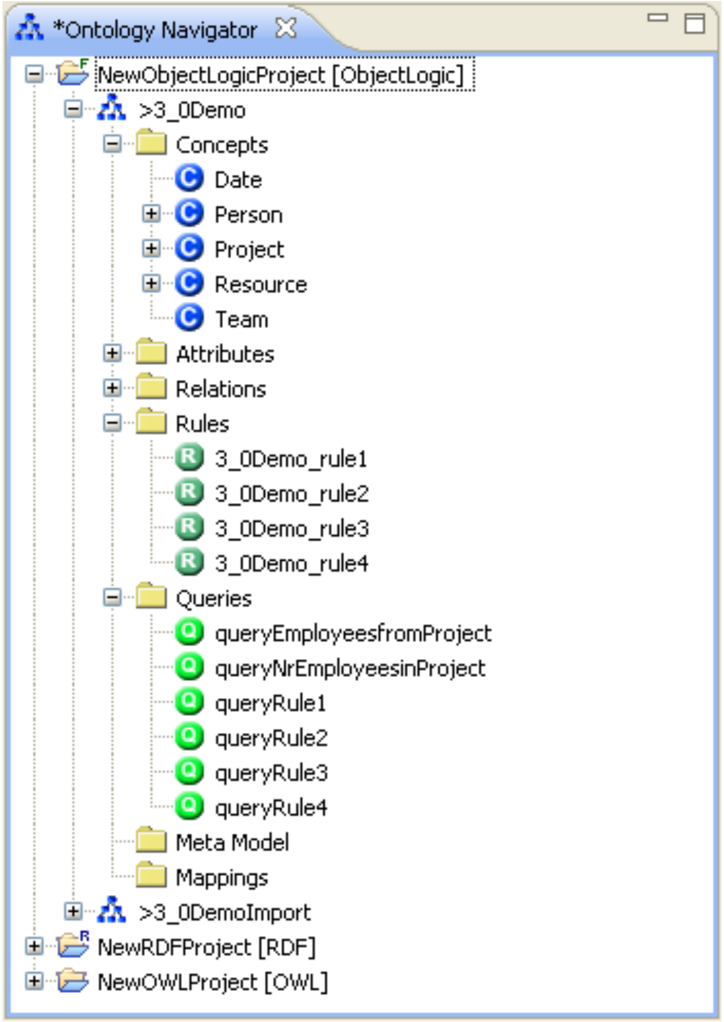
Corresponds to

```
has_son(X,Y) :- man(Y), has_father(Y,X).
```

ONTOSTUDIO

http://www.semafora-systems.com/en/products/ontostudio/download/

OntoStudio

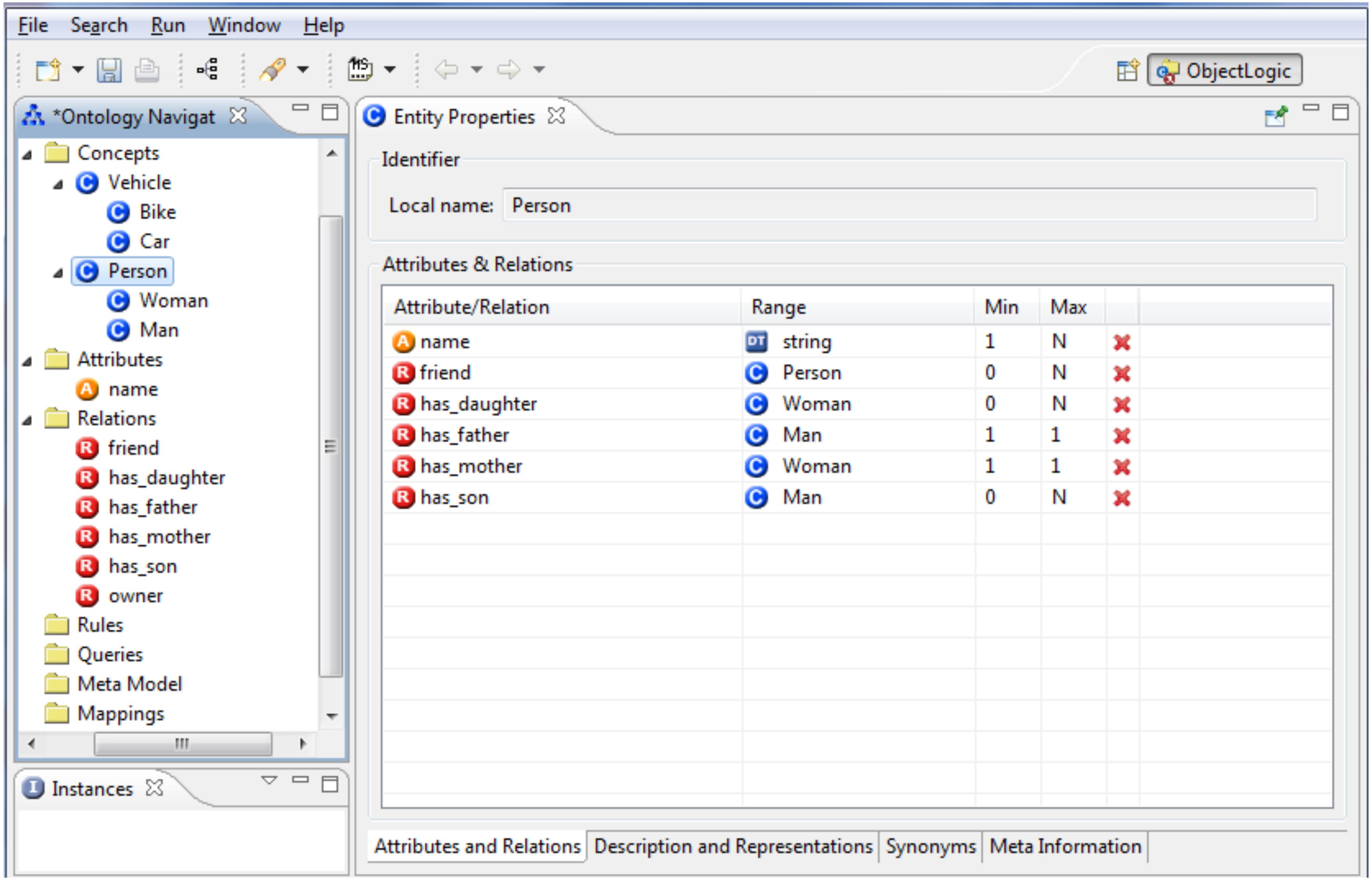


*Ontology Navigator

- NewObjectLogicProject [ObjectLogic]
  - >3_0Demo
    - Concepts
      - Date
      - Person
      - Project
      - Resource
      - Team
    - Attributes
    - Relations
    - Rules
      - 3_0Demo_rule1
      - 3_0Demo_rule2
      - 3_0Demo_rule3
      - 3_0Demo_rule4
    - Queries
      - queryEmployeesfromProject
      - queryNrEmployeesinProject
      - queryRule1
      - queryRule2
      - queryRule3
      - queryRule4
    - Meta Model
    - Mappings
  - >3_0DemoImport
- NewRDFProject [RDF]
- NewOWLProject [OWL]

- OntoStudio is a tool from Ontoprise GmbH. It supports graphical representation of
  - OWL
  - RDF
  - ObjectLogic

ObjectLogic Perspective in OntoStudio: Defining Objects and Attributes/Relations



- Concepts
  - Vehicle
    - Bike
    - Car
  - Person
    - Woman
    - Man
- Attributes
  - name
- Relations
  - friend
  - has_daughter
  - has_father
  - has_mother
  - has_son
  - owner
- Rules
- Queries
- Meta Model
- Mappings

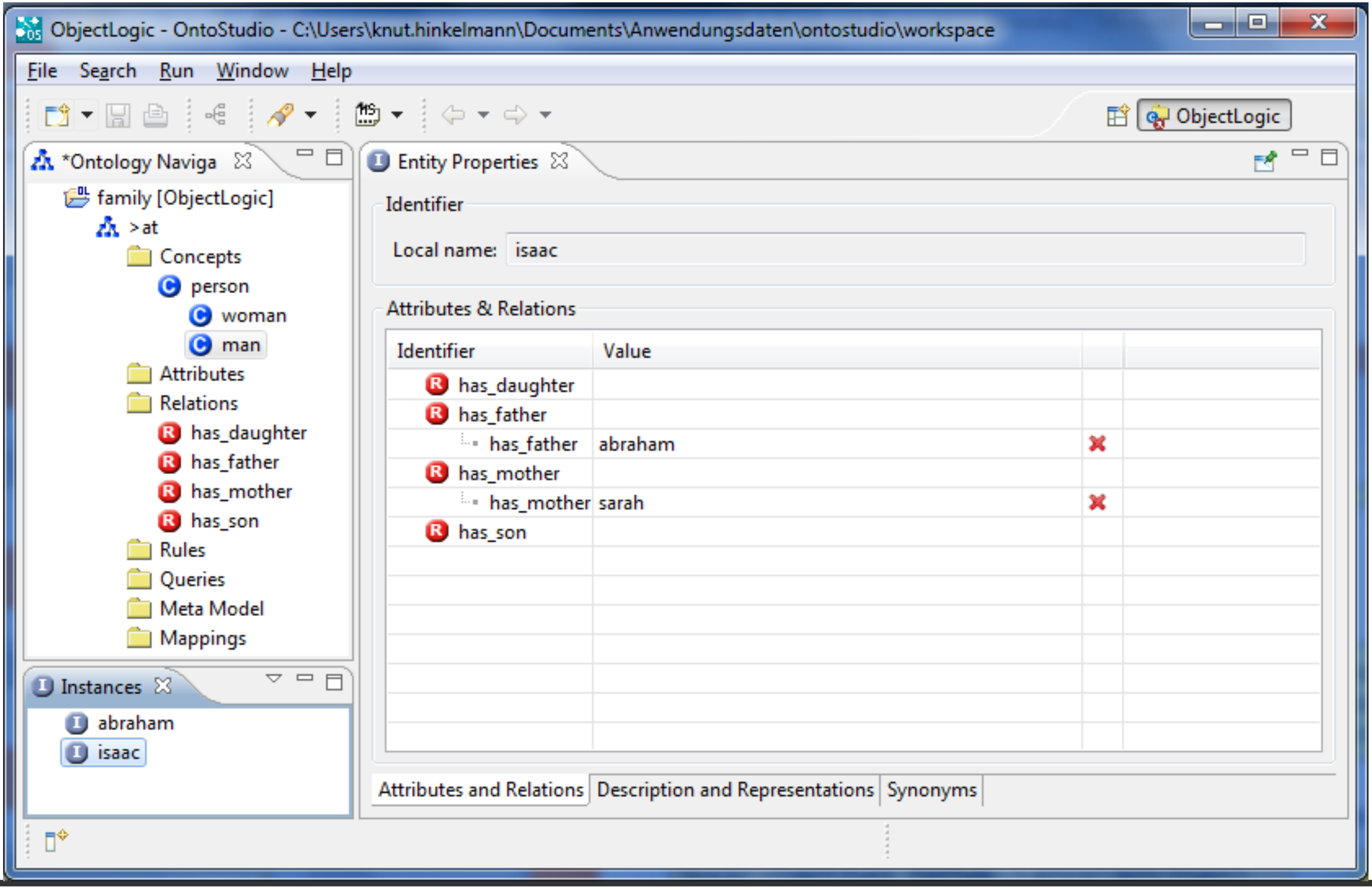
Identifier

Local name: Person

Attributes & Relations

Attribute/Relation	Range	Min	Max	
A name	DT string	1	N	✖
R friend	C Person	0	N	✖
R has_daughter	C Woman	0	N	✖
R has_father	C Man	1	1	✖
R has_mother	C Woman	1	1	✖
R has_son	C Man	0	N	✖

Instances and Relation Values


ObjectLogic - OntoStudio - C:\Users\knut.hinkelmann\Documents\Anwendungsdaten\ontostudio\workspace
File
Search
Run
Window
Help
ObjectLogic
*Ontology Naviga
Entity Properties
Instances
Attributes and Relations
Description and Representations
Synonyms

- family [ObjectLogic]
  - >at
    - Concepts
      - person
        - woman
        - man
    - Attributes
    - Relations
      - has_daughter
      - has_father
      - has_mother
      - has_son
    - Rules
    - Queries
    - Meta Model
    - Mappings

Instances:

- abraham
- isaac

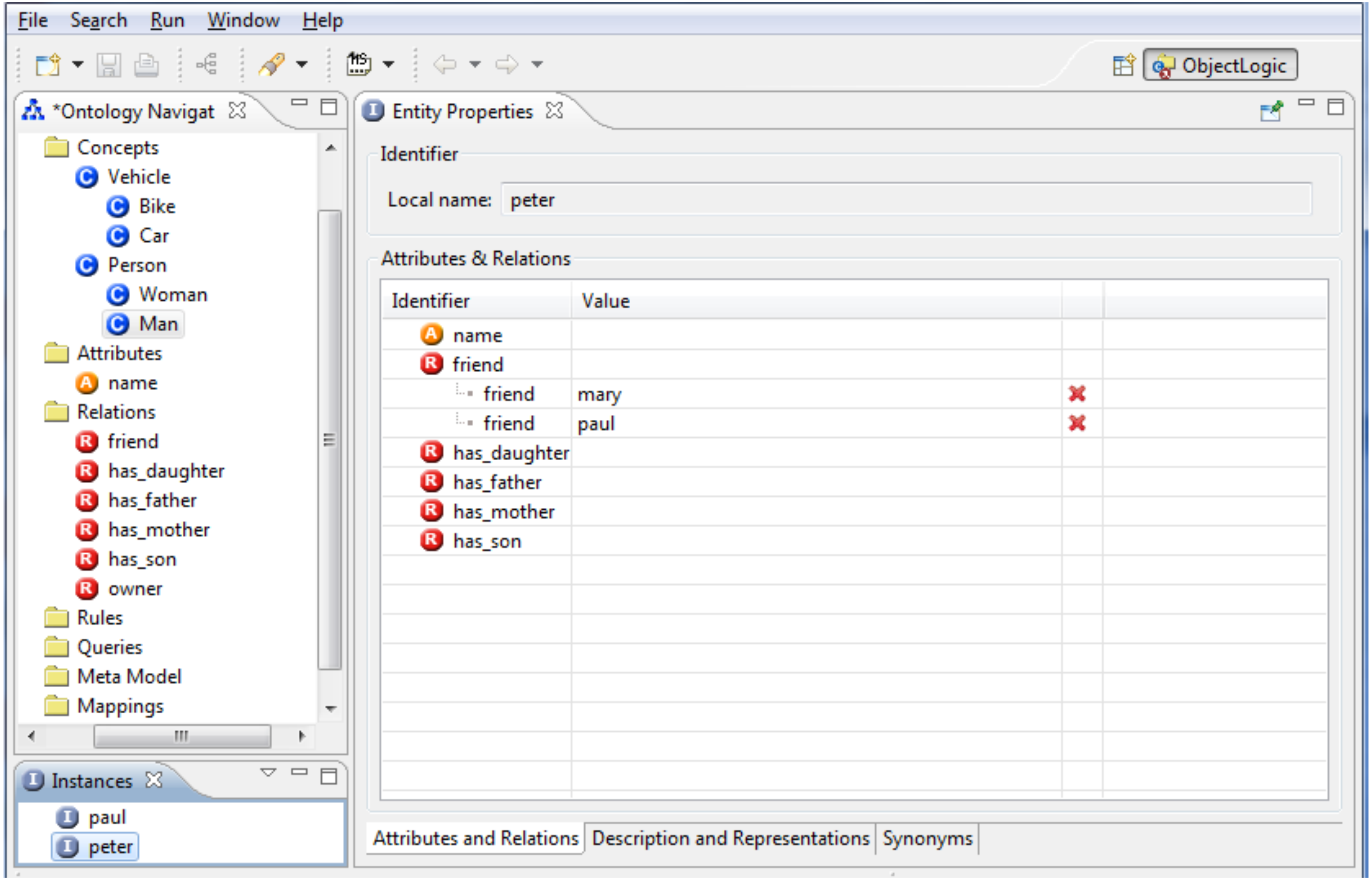
Identifier

Local name: isaac

Attributes & Relations

Identifier	Value	
has_daughter		
has_father		
has_father	abraham	✖
has_mother		
has_mother	sarah	✖
has_son		

Instances and Relation Values



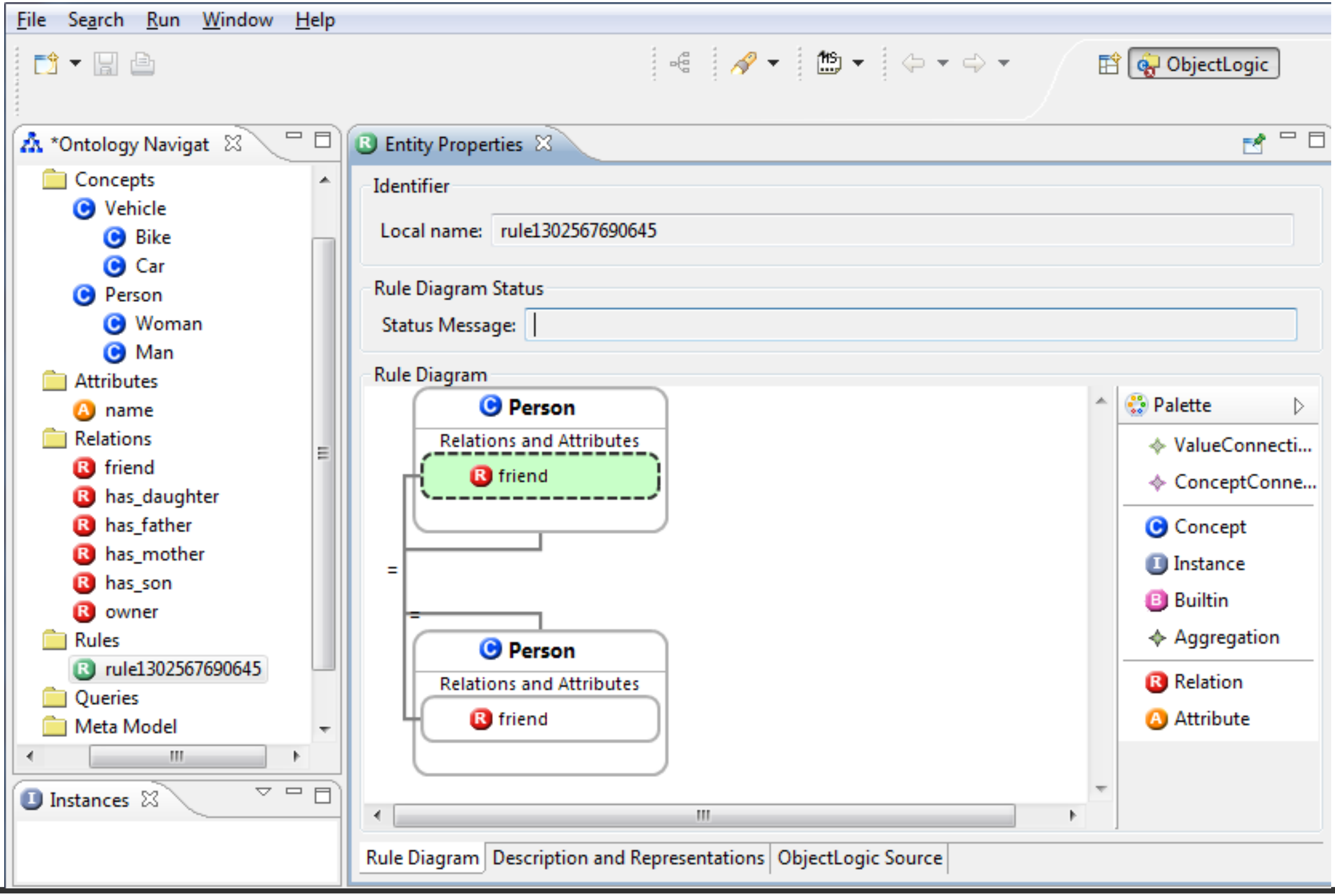
Identifier

Local name: peter

Attributes & Relations

Identifier	Value	
name		
friend		
friend	mary	✖
friend	paul	✖
has_daughter		
has_father		
has_mother		
has_son		

Rule Diagrams



Identifier

Local name: rule1302567690645

Rule Diagram Status

Status Message: