Web Ontology Language OWL – Part I

VL Semantic Technologies

Bernd Neumayr

(with contributions from Dieter Steiner)

Department for Business Informatics – Data & Knowledge Engineering

based on: OWL 2 Web Ontology Language Primer (Second Edition), W3C Recommendation 11 December 2012
http://www.w3.org/TR/owl2-primer/

OWL 2 Part I – Agenda

JKU

1. Introduction
2. Modeling with Classes, Properties, and Individuals
3. Class Expressions

Introduction

- What is an Ontology?
- What is OWL?

What is an Ontology?

An ontology is a formal, explicit specification of a shared conceptualization.

[Studer et al., 1998]

Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among agents. Ontological commitments are agreements to use the shared vocabulary in a coherent and consistent manner.

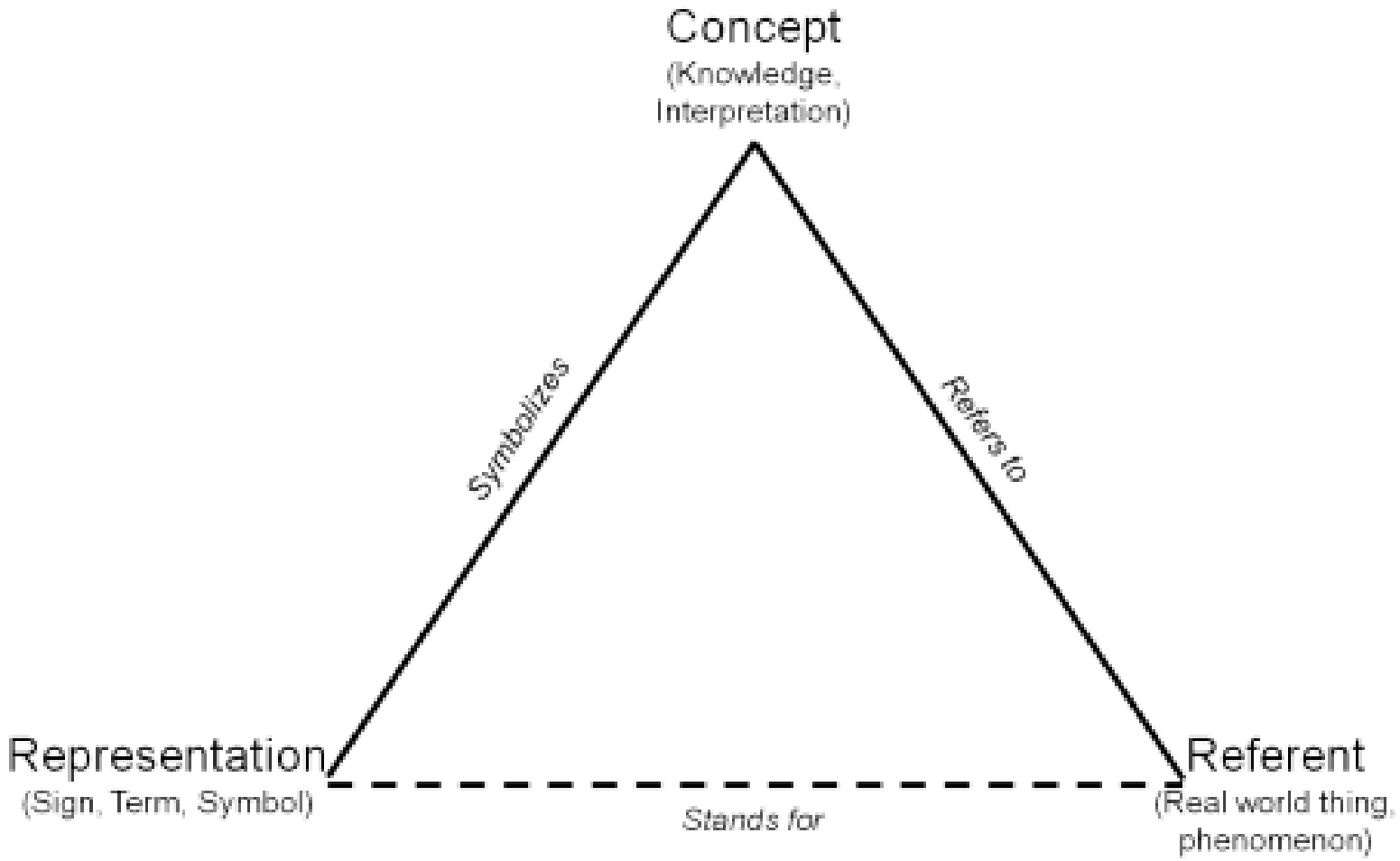
[Gruber, 1995]

What is an Ontology?

- Ontologies are formal models of an application domain which facilitate exchanging and sharing of knowledge.
- From a methodical point of view, techniques of object-oriented modelling are developed further so that models are not only used to structure software, but also represent an explicit element of the user interface and are used at runtime.
- From a socio-cultural point of view, ontologies require mutual consent of a group of users regarding the respective terms and their relationships.

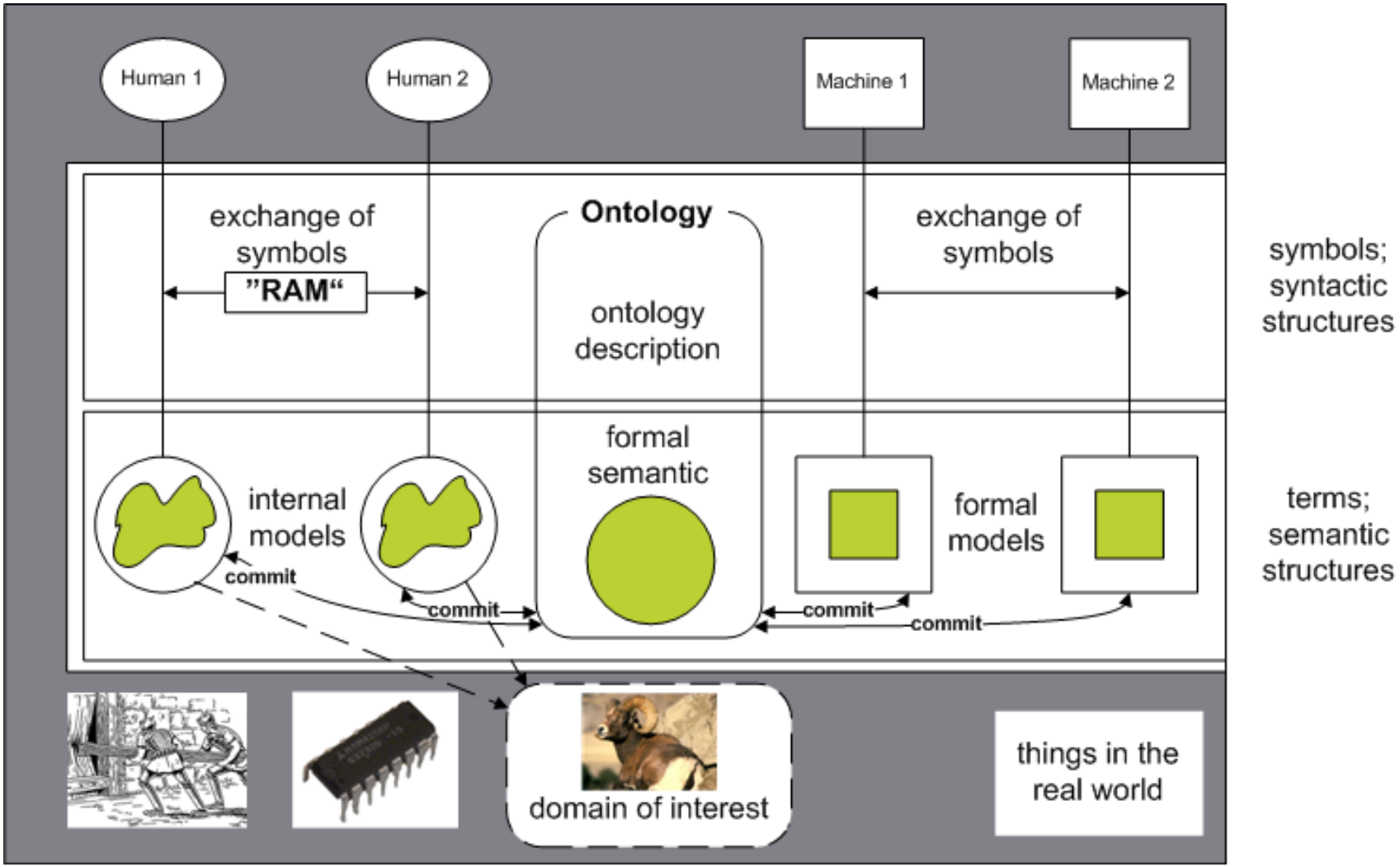
Source: [Maedche et al., 2001] (Recommended Reading)

Semiotic Triangle



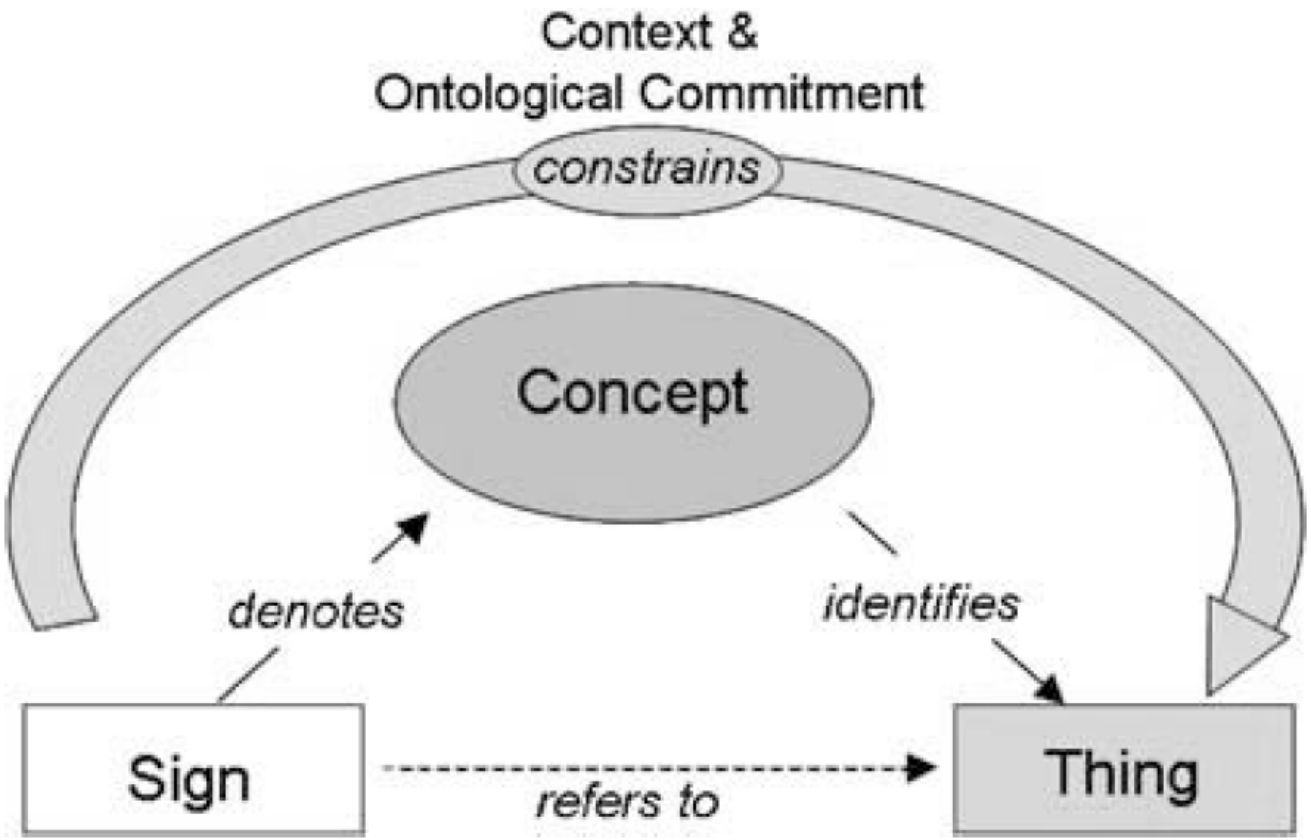
Source: `http://activeknowledgemodeling.com/2010/12/17/`

Levels of Communication



Source: [Maedche et al., 2001]

Semiotic Triangle Revisited



Source: [Guarino et al., 2009]

OWL 2 Web Ontology Language

- OWL 2 is a language for expressing ontologies
- here, an ontology is
  - a computational artifact,
  - a document,
  - a set of precise descriptive statements about a part of the world (domain of interest, subject matter)
- precise descriptions have several purposes
  - prevent misunderstandings in human communication
  - ensure that software behaves in a uniform way and works well with other software

Terminological and Assertional Knowledge

JKU

An OWL ontology consists of

- Terminological Knowledge (TBox)
  - Vocabulary: a set of central terms (classes, properties)
  - Meaning of terms: natural language descriptions
  - + formal description of interrelations between terms
- Assertional Knowledge (ABox)
  - concrete objects (individuals) of the considered domain
  - relationships between individuals
  - properties of individuals

OWL 2 – W3C Recommendations

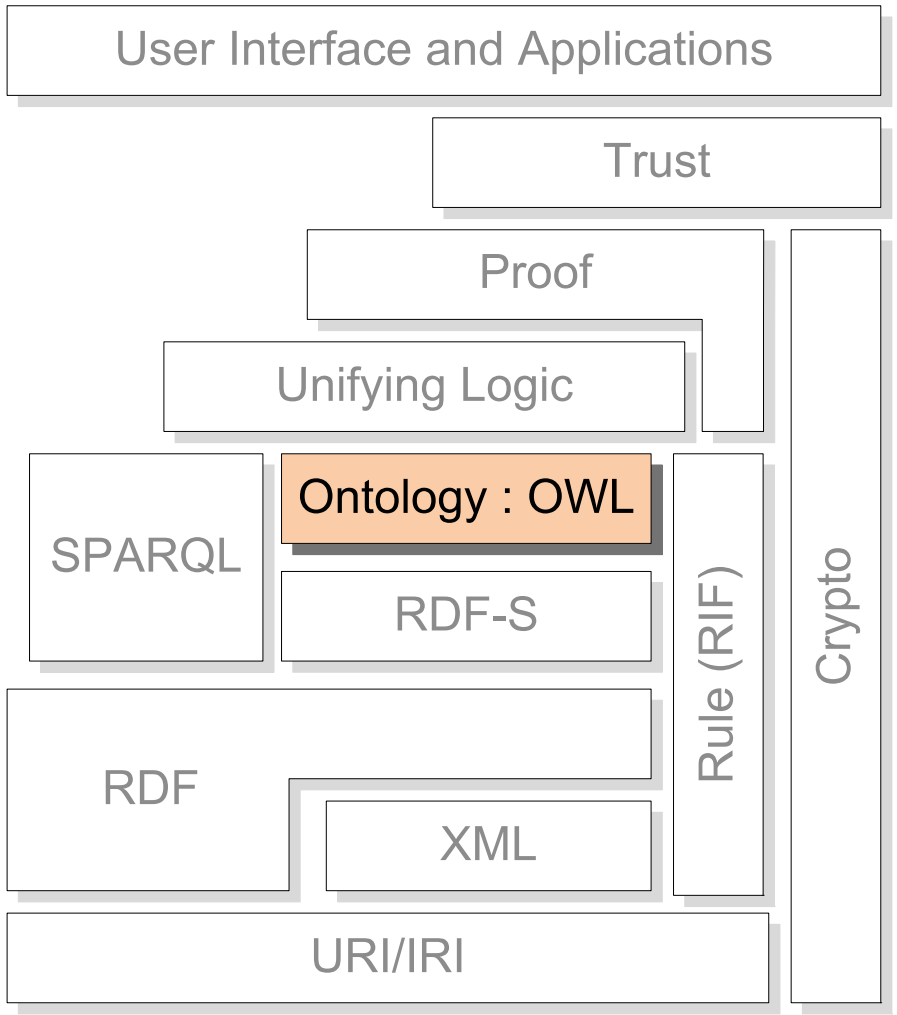
these slides are based on

- OWL 2 Web Ontology Language Primer
  `http://www.w3.org/TR/owl2-primer/`

overview of recommendations:

- OWL 2 Document Overview
  `http://www.w3.org/TR/owl2-overview/`

OWL and the Semantic Web Stack



OWL is not a programming language

- OWL is declarative: describe state of affairs in logical way
- Appropriate tools (reasoners) infer further information about state of affairs
- How reasoning works algorithmically is not part of OWL
- What inferences to be made is predetermined by formal semantics:
  - OWL 2 Direct Semantics
  - OWL 2 RDF-Based Semantics
- reuse of ideas from software engineering: methodological and collaborative aspects, modularization, patterns, etc.

OWL is not a schema language

- OWL 2 is not a schema language for syntax conformance.
- no means to prescribe how a document should be structured
- no way to enforce that a certain piece of information (like the social security number of a person) has to be syntactically present

OWL is not a database framework

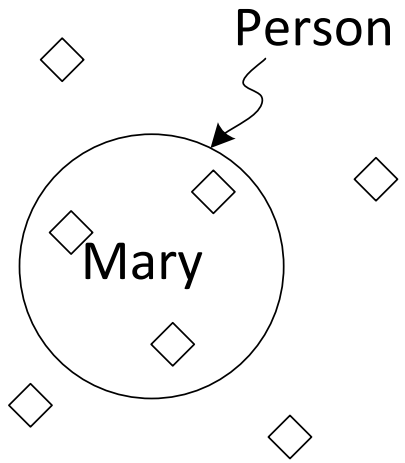
OWL document	Database
terminological knowledge	database schema
assertional knowledge	database content
open world assumption (missing statement = unknown)	closed world assumption (missing statement = false)
“descriptive” constraints	“prescriptive” constraints

Modeling with Classes, Properties, and Individuals

Classes and Instances

- Classes represent sets of individuals.
- Typically, a class (e.g., `Person`) denotes the set of objects comprised by a concept of human thinking, like the concept *person*.

```
Class: Person

Individual: Mary
    Types: Person
```



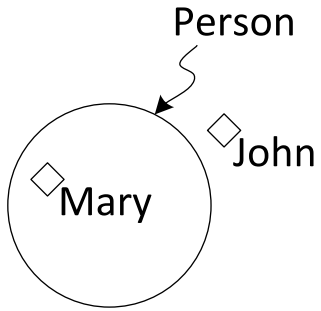
Open World

```
Class: Person

Individual: Mary
    Types: Person

Individual: John
```

Open World

```
Class: Person

Individual: Mary
    Types: Person

Individual: John
```

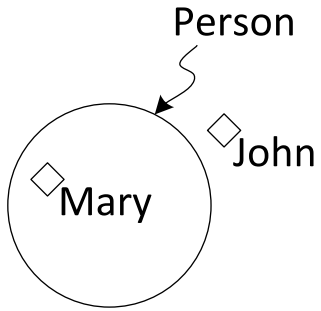
Interpretation with closed world and unique name assumption:



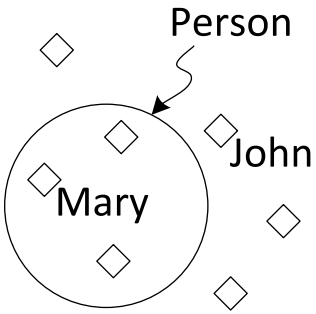
Open World

```
Class: Person

Individual: Mary
    Types: Person

Individual: John
```

Interpretation with closed world and unique name assumption:



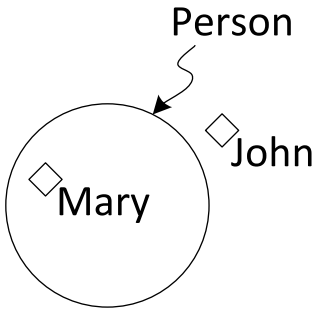
Possible interpretations with open world assumption and without unique name assumption:



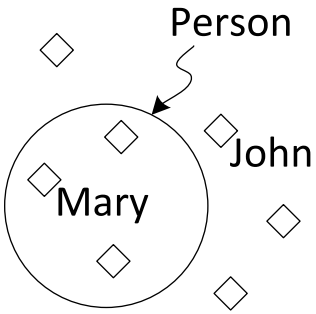
Open World

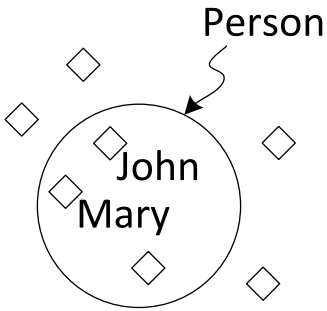
```
Class: Person

Individual: Mary
    Types: Person

Individual: John
```

Interpretation with closed world and unique name assumption:



Possible interpretations with open world assumption and without unique name assumption:

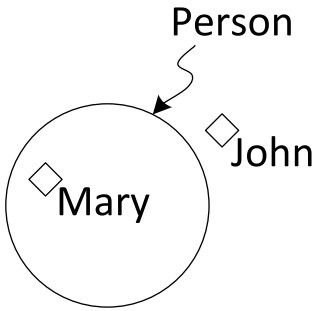




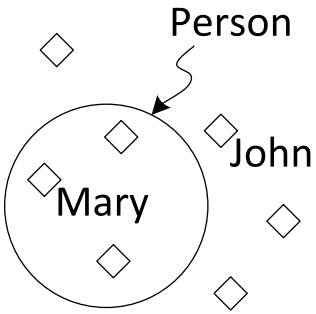
Open World

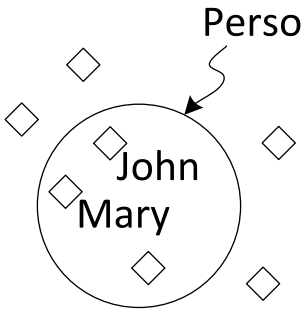
```
Class: Person

Individual: Mary
    Types: Person

Individual: John
```

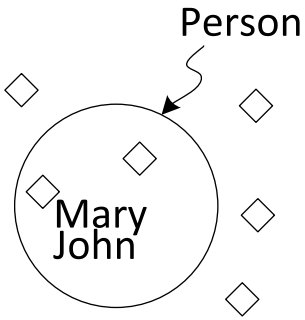
Interpretation with closed world and unique name assumption:



Possible interpretations with open world assumption and without unique name assumption:





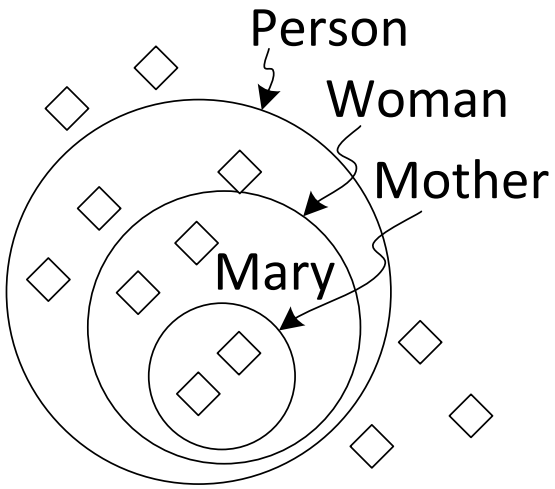


...

Class Hierarchies

"Every *mother* is a *woman*." "Every *woman* is a *person*." "*Mary* is a *woman*."

```
Class: Person
Class: Woman
    SubClassOf: Person
Class: Mother
    SubClassOf: Woman
Individual: Mary
    Types: Woman
```

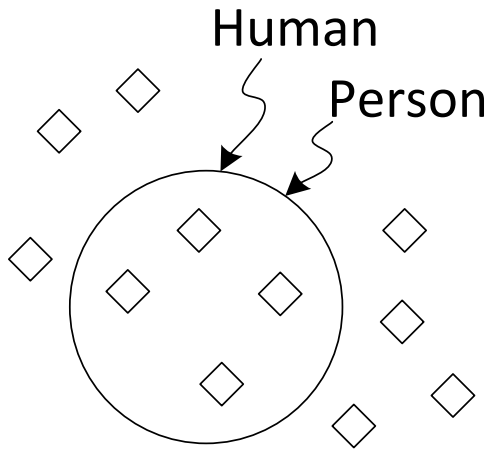


Consequences: "Every *mother* is a *person*." "*Mary* is a *person*."

Equivalent Classes

We use *human* and *person* interchangeably.

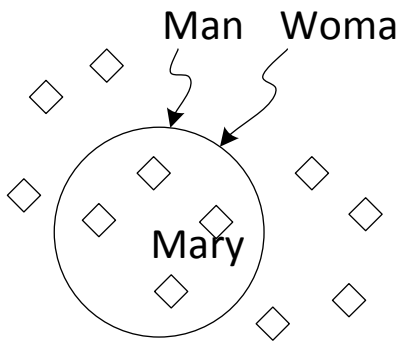
```
Class: Human

Class: Person
    EquivalentTo: Human
```

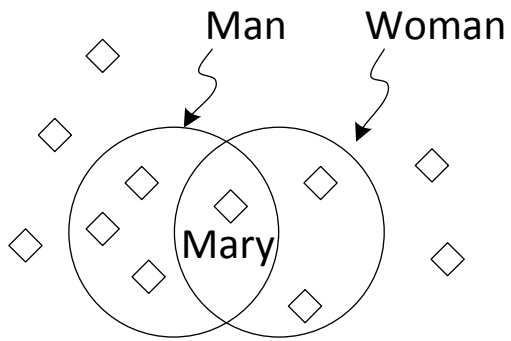


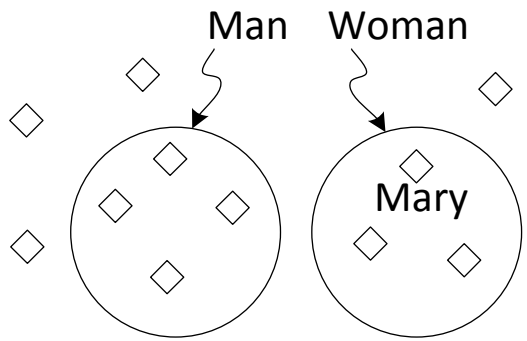
Disjoint Classes

```
Class: Woman
Class: Man
Individual: Mary
    Types: Woman
```

Some possible interpretations:



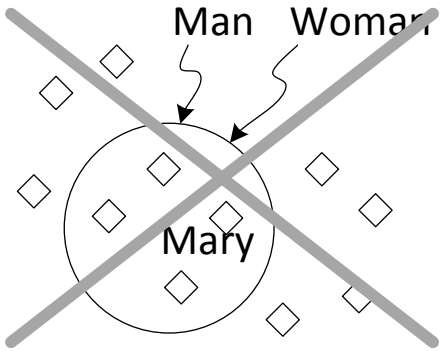


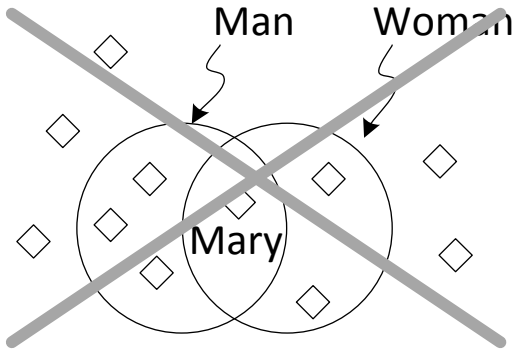


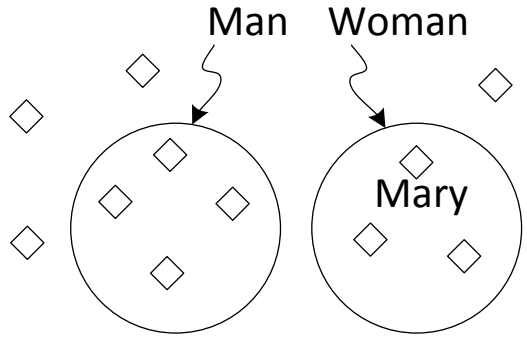
Disjoint Classes

```
Class: Woman
Class: Man
Individual: Mary
    Types: Woman
DisjointClasses: Woman, Man
```

Some possible interpretations:



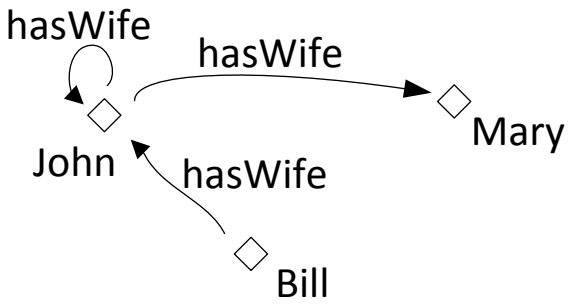


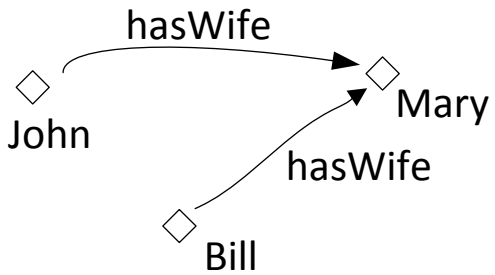


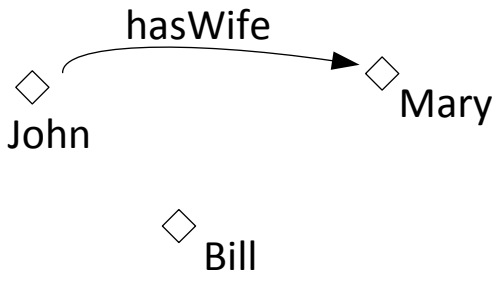
Object Properties

```
ObjectProperty: hasWife
Individual: John
    Facts: hasWife Mary
Individual: Bill
```

Some possible interpretations:



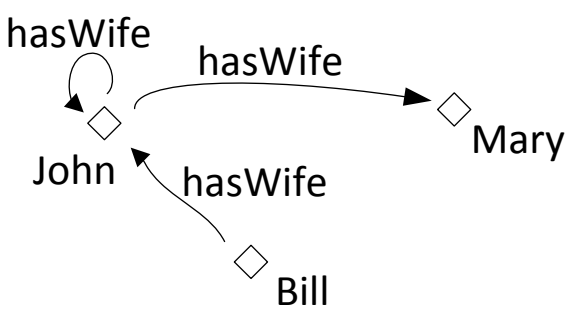


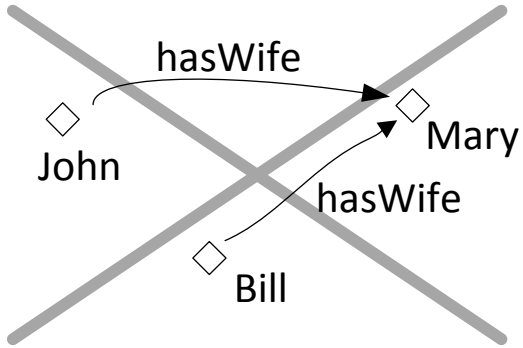


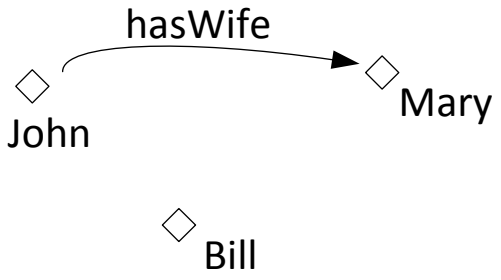
Object Properties

```
ObjectProperty: hasWife
Individual: John
    Facts: hasWife Mary
Individual: Bill
    Facts: not hasWife Mary
```

Some possible interpretations:







Property Hierarchies
known from RDF Schema

```
ObjectProperty: hasWife
  SubPropertyOf: hasSpouse
```

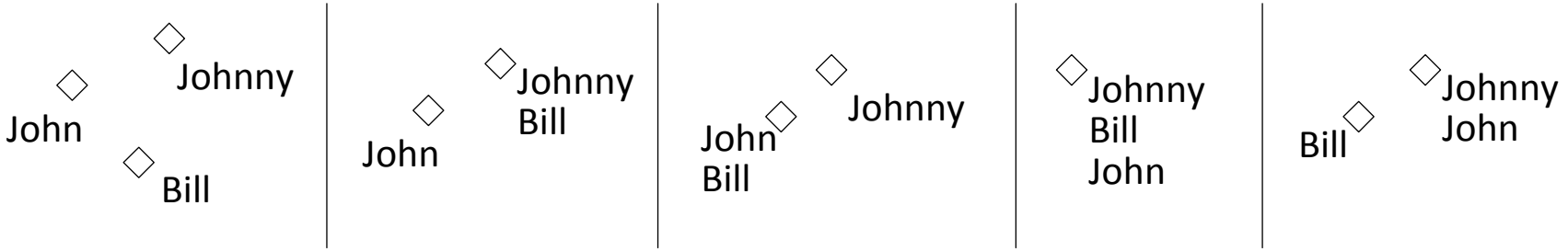
Domain and Range

```
ObjectProperty: hasWife
   Domain: Man
   Range: Woman
```

Identity

```
Individual: John

Individual: Bill

Individual: Johnny
```

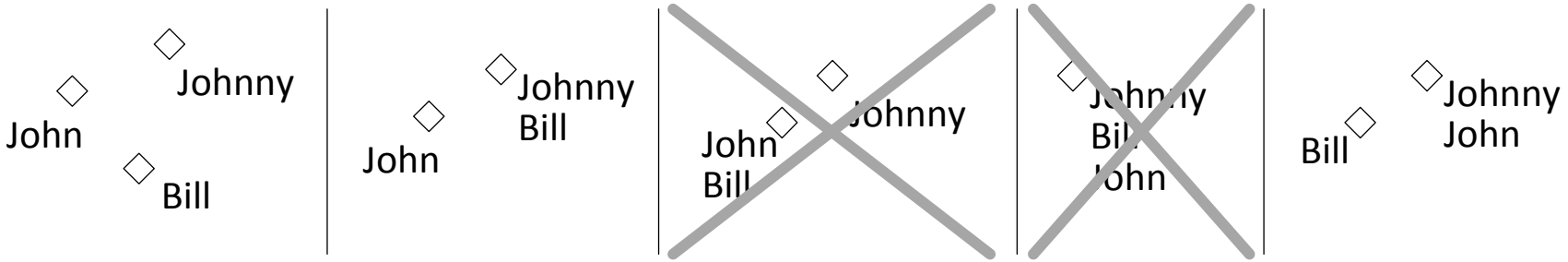
Some possible interpretations:



Identity

```
Individual: John
    DifferentFrom: Bill
Individual: Bill

Individual: Johnny
```

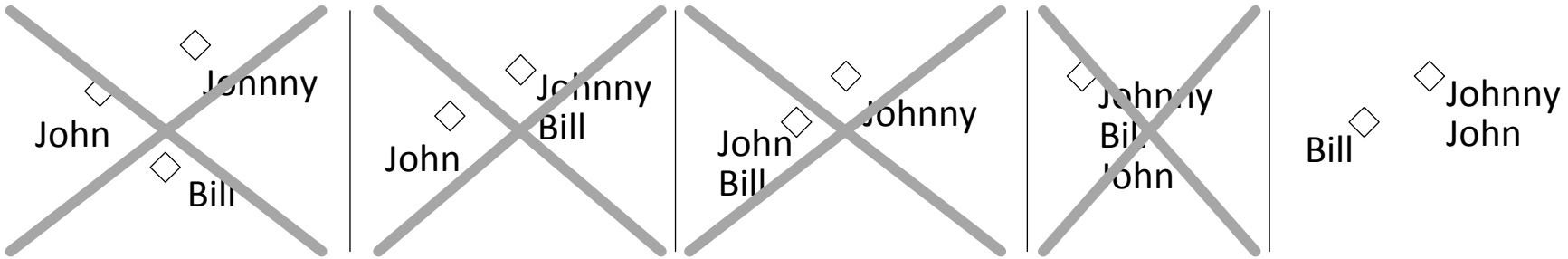
Some possible interpretations:



Identity

```
Individual: John
    DifferentFrom: Bill
Individual: Bill

Individual: Johnny
    SameAs: John
```

Some possible interpretations:



Datatypes

```
Individual: John
    Facts: hasAge "51"^^xsd:integer

Individual: Jack
    Facts: not hasAge "53"^^xsd:integer

DataProperty: hasAge
    Domain: Person
    Range:  xsd:nonNegativeInteger
```

Class Expressions

- Complex Classes
- Property Restrictions
- Property Cardinality Restrictions
- Enumeration of Individuals

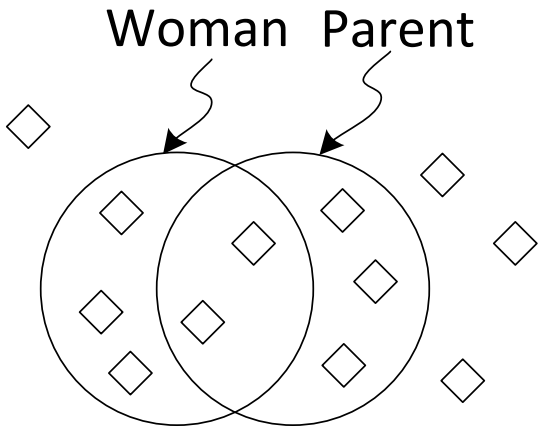
Class Expressions

Class expressions combine named classes, properties, and individuals and may be used where named classes are used:

- necessary and sufficient conditions
  `EquivalentTo: ...`
- necessary conditions
  `SubClassOf: ...`
- class assertions
  `Types: ...`
- ...

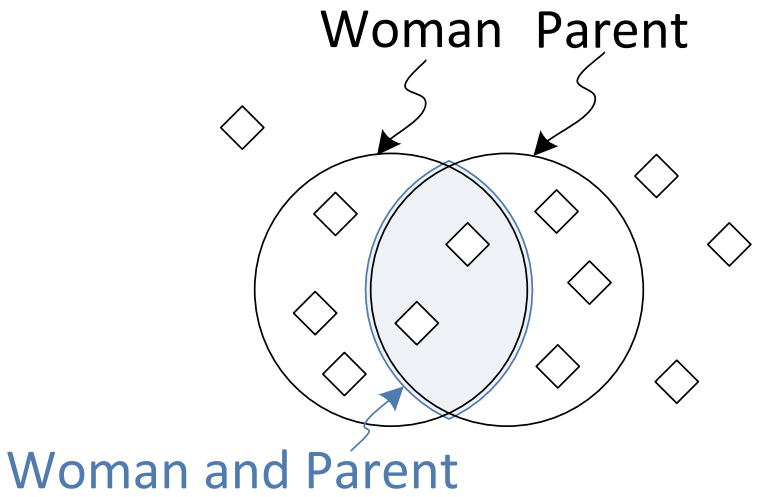
Intersection

The concept ‘Mother’.



Intersection

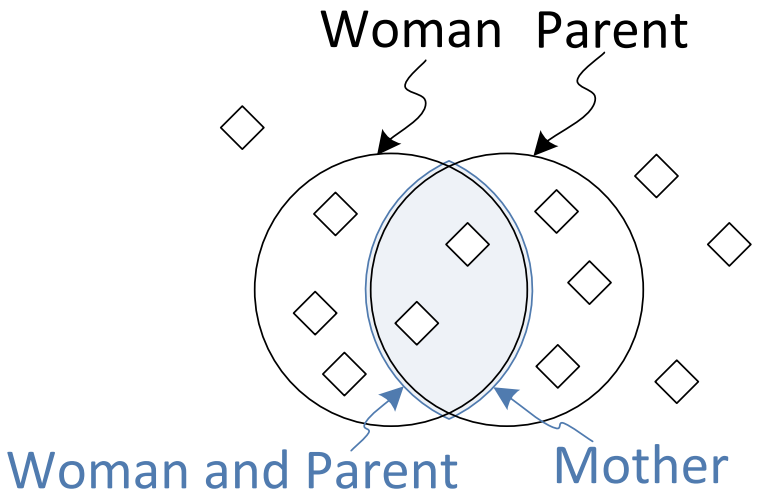
The concept ‘Mother’.



Intersection

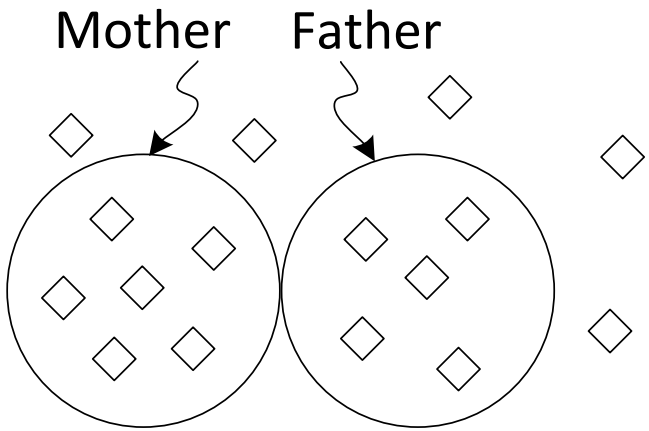
The concept ‘Mother’.

```
Class: Mother
  EquivalentTo: Woman and Parent
```



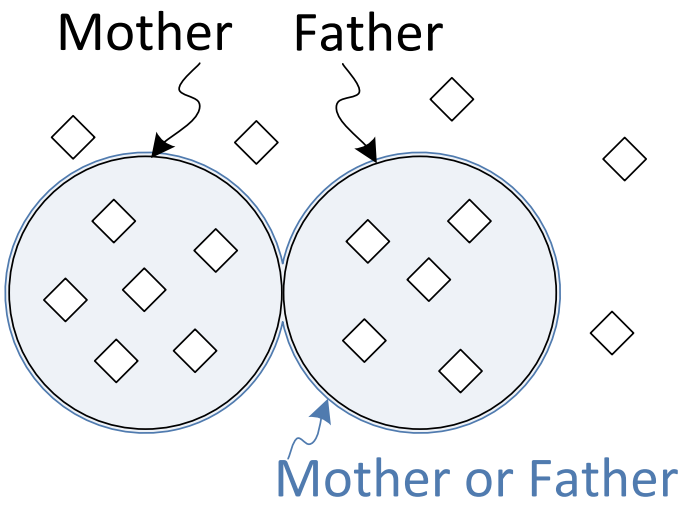
Union

The concept ‘Parent’.



Union

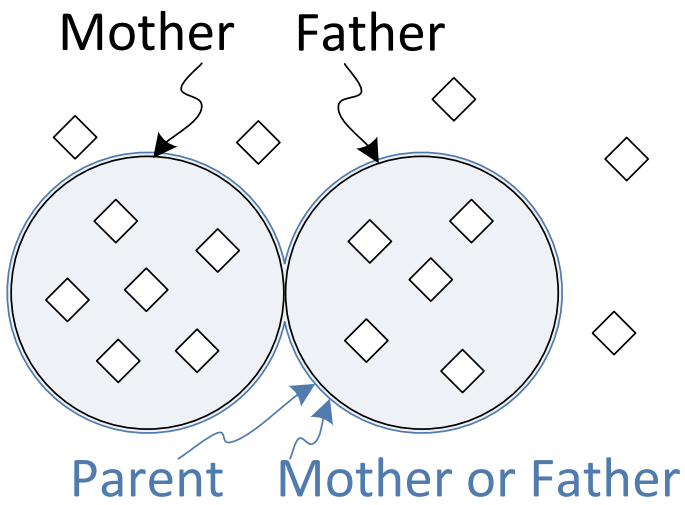
The concept ‘Parent’.



Union

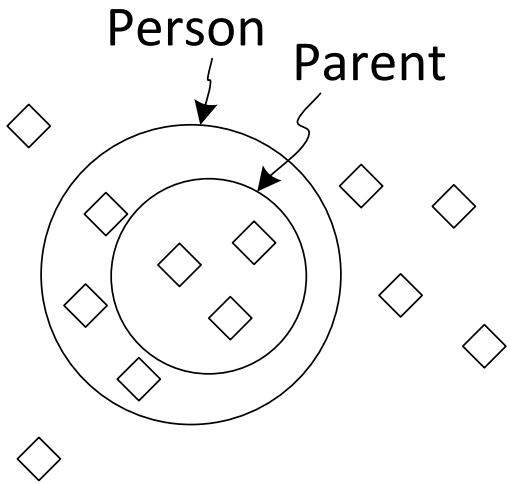
The concept ‘Parent’.

```
Class: Parent
  EquivalentTo: Mother or Father
```



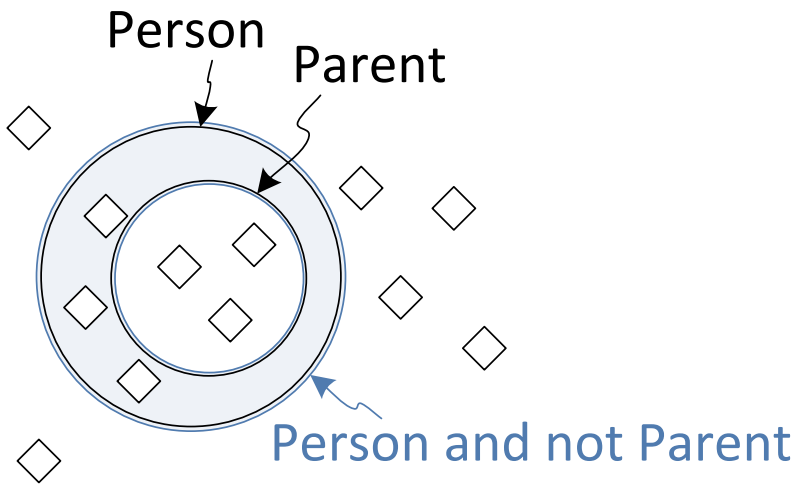
Complement

The concept ‘Person without Child’:



Complement

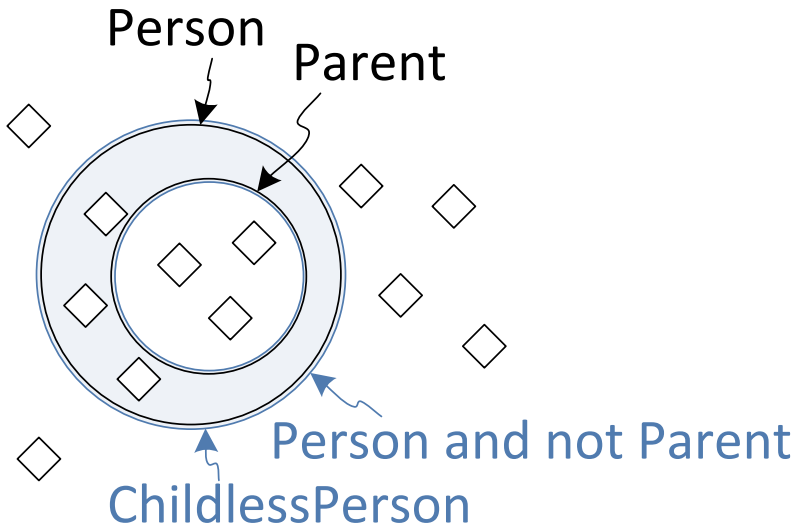
The concept ‘Person without Child’:



Complement

The concept ‘Person without Child’:

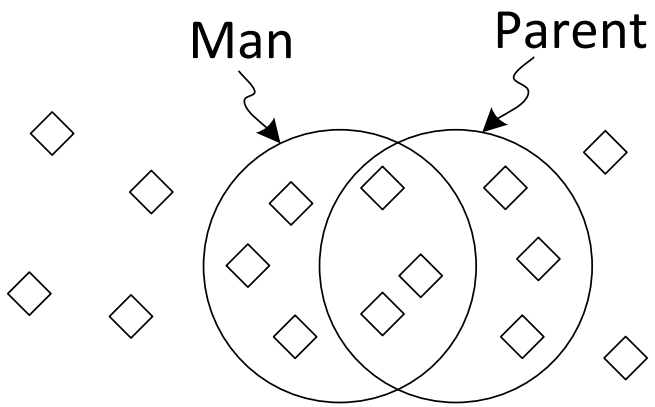
```
Class: ChildlessPerson
  EquivalentTo: Person and not Parent
```



Necessary Conditions

'Every grandfather is (necessarily) a man and a parent.'

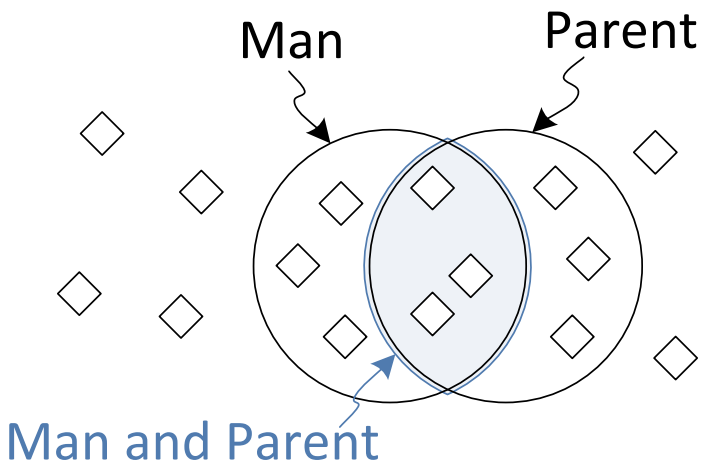
```
Class: Grandfather
  SubClassOf: Man and Parent
```



Necessary Conditions

'Every grandfather is (necessarily) a man and a parent.'

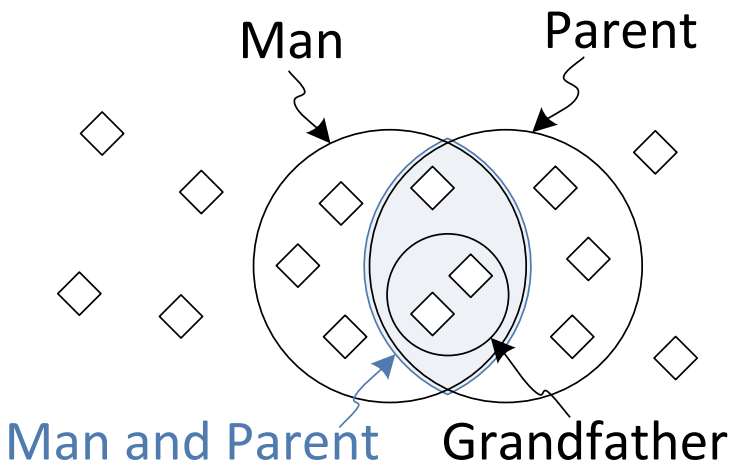
```
Class: Grandfather
  SubClassOf: Man and Parent
```



Necessary Conditions

'Every grandfather is (necessarily) a man and a parent.'

```
Class: Grandfather
  SubClassOf: Man and Parent
```



Class Expression in Class Assertions

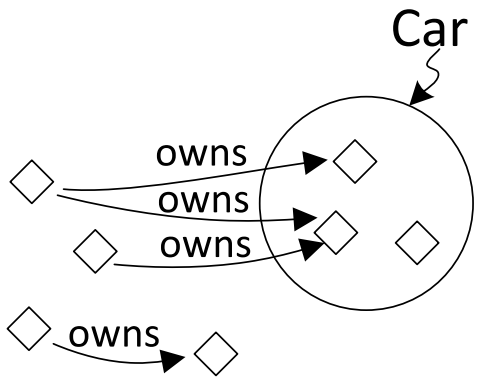
JKU

'Jack is a Person but not a Parent.'

```
Individual: Jack
  Types: Person and not Parent
```

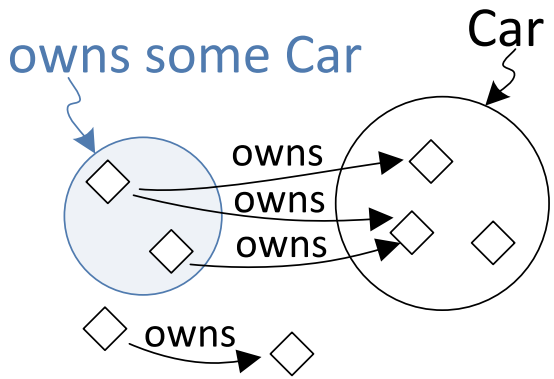
Existential Quantification

The concept ‘Car Owner’:



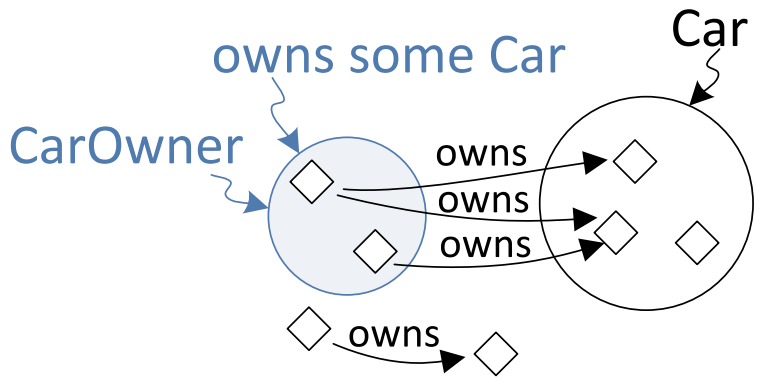
Existential Quantification

The concept ‘Car Owner’:



Existential Quantification

The concept ‘Car Owner’:

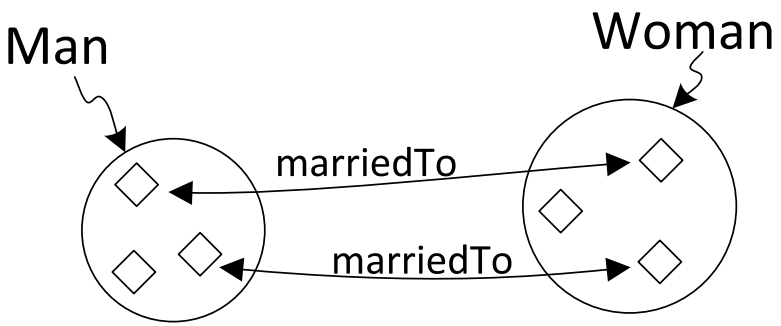



```
Class: CarOwner
    EquivalentTo: owns some Car
```

Universal Quantification

Necessary Condition

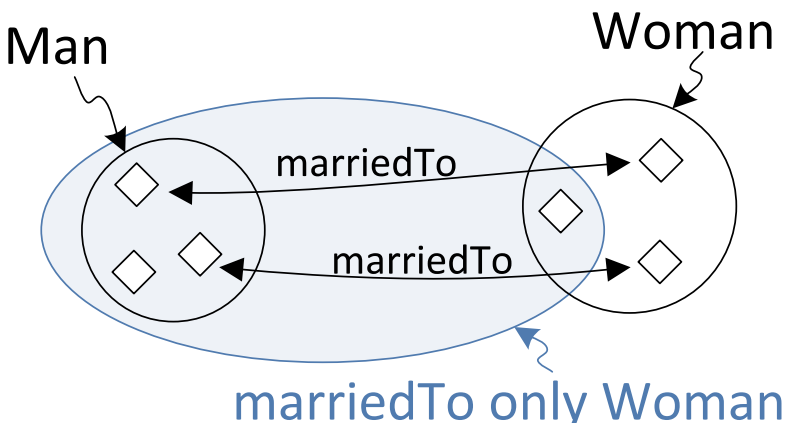
"Men only marry Women."



Universal Quantification

Necessary Condition

"Men only marry Women."

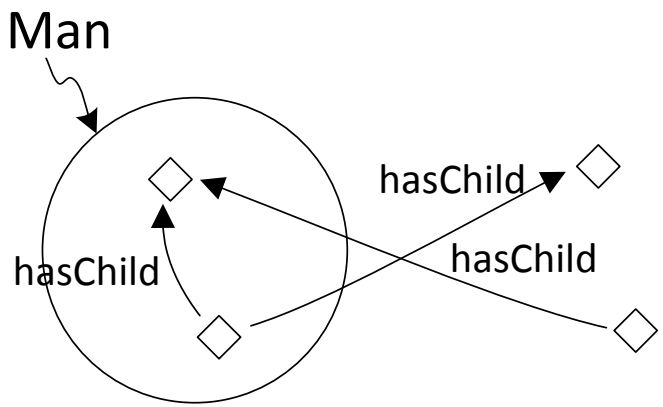



```
Class: Man
    SubClassOf: marriedTo only Woman
```

Existential and Universal Quantification

Necessary and Sufficient Condition

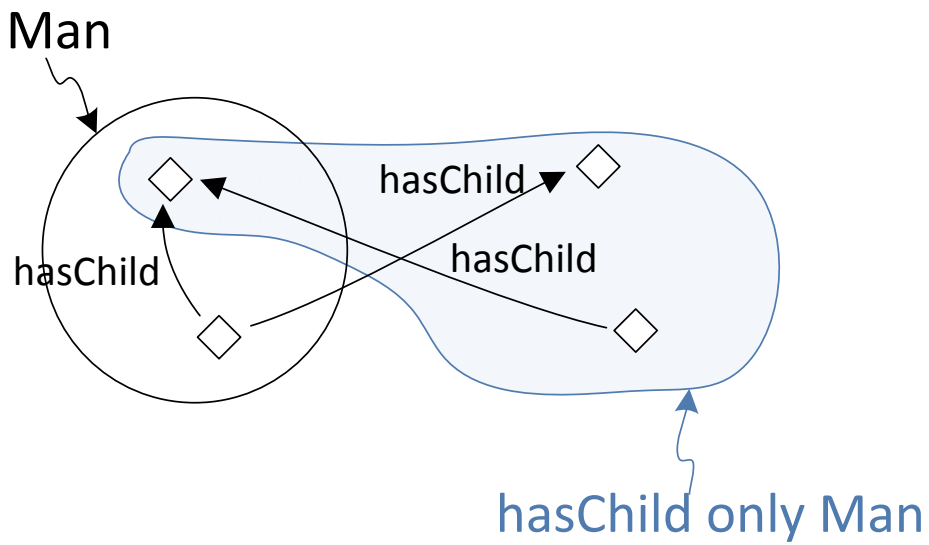
```
Class: ParentOfOnlySons
    EquivalentTo: hasChild some Man
                  and hasChild only Man
```



Existential and Universal Quantification

Necessary and Sufficient Condition

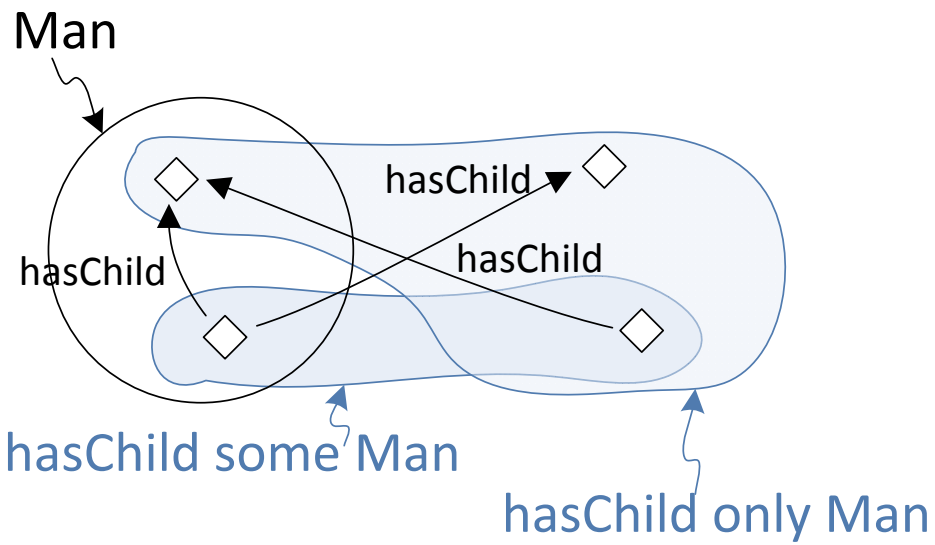
```
Class: ParentOfOnlySons
    EquivalentTo: hasChild some Man
                  and hasChild only Man
```



Existential and Universal Quantification

Necessary and Sufficient Condition

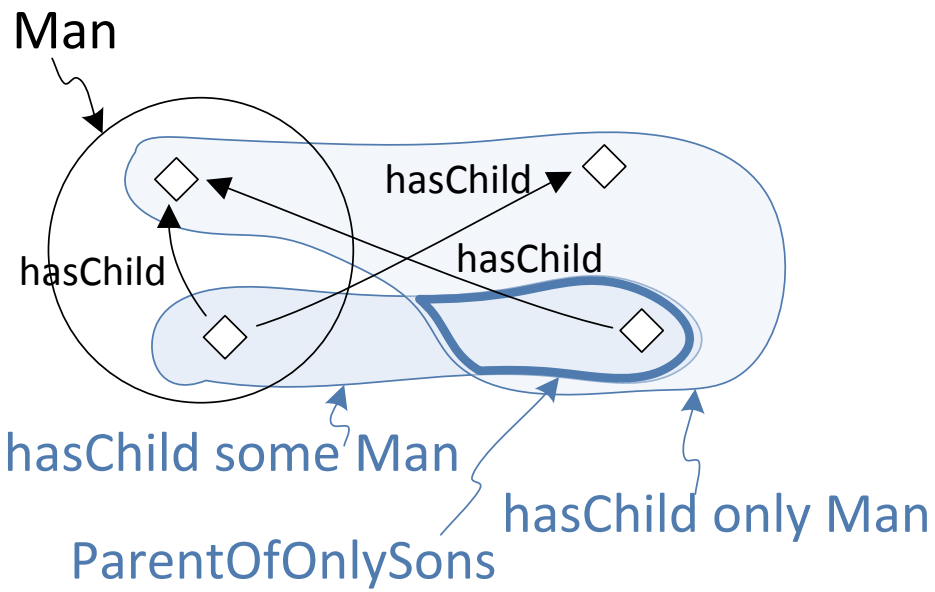
```
Class: ParentOfOnlySons
    EquivalentTo: hasChild some Man
                  and hasChild only Man
```



Existential and Universal Quantification

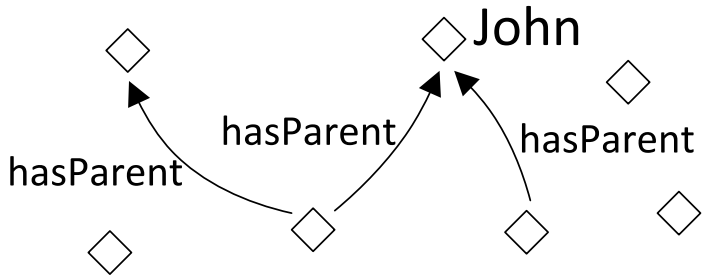
Necessary and Sufficient Condition

```
Class: ParentOfOnlySons
    EquivalentTo: hasChild some Man
                  and hasChild only Man
```



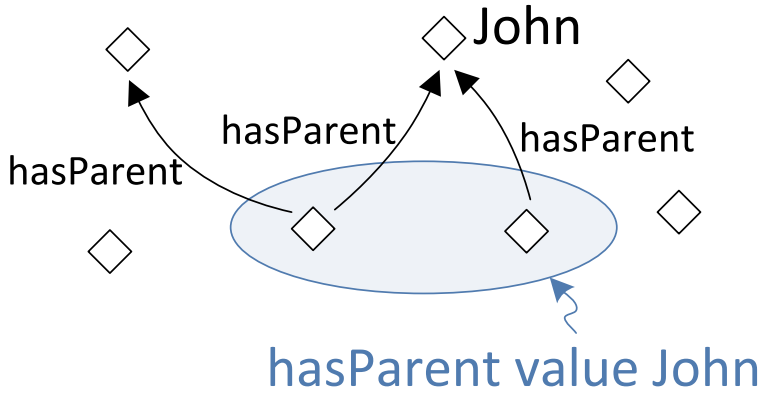
Individual Value Restriction

```
Class: JohnsChildren
  EquivalentTo: hasParent value John
```



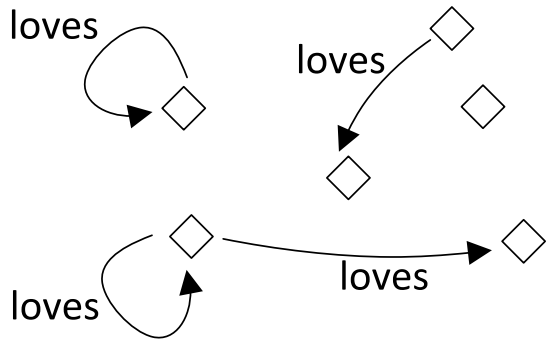
Individual Value Restriction

```
Class: JohnsChildren
  EquivalentTo: hasParent value John
```



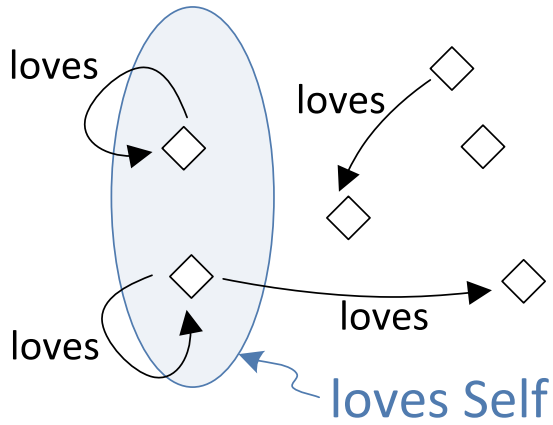
Self Restriction

```
Class: NarcisticPerson
    EquivalentTo: loves Self
```



Self Restriction

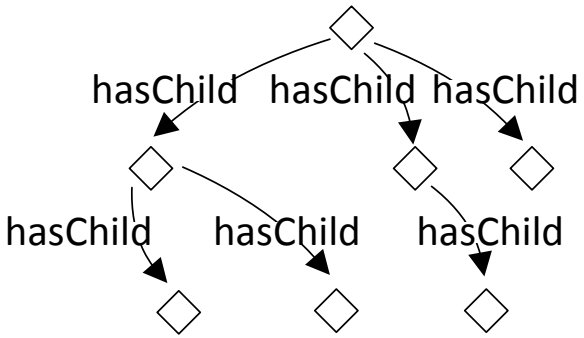
```
Class: NarcisticPerson
    EquivalentTo: loves Self
```



Property Cardinality Restrictions

"Individuals with

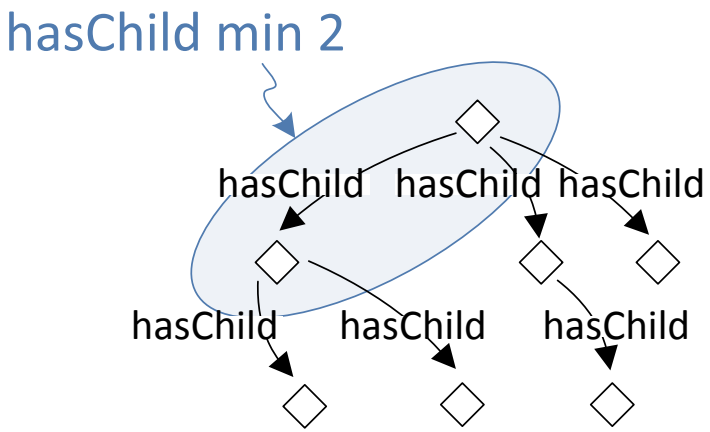
- at least two children"
- at most two children"
- exactly two children"
- at least two children that are parents themselves"



Property Cardinality Restrictions

"Individuals with

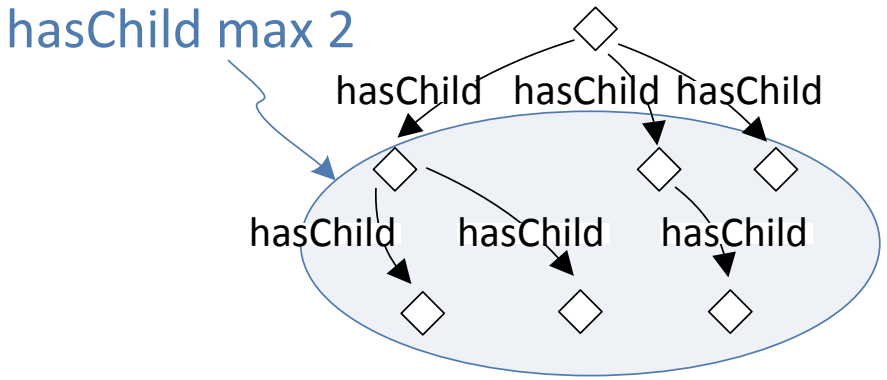
- at least two children"
- at most two children"
- exactly two children"
- at least two children that are parents themselves"



Property Cardinality Restrictions

"Individuals with

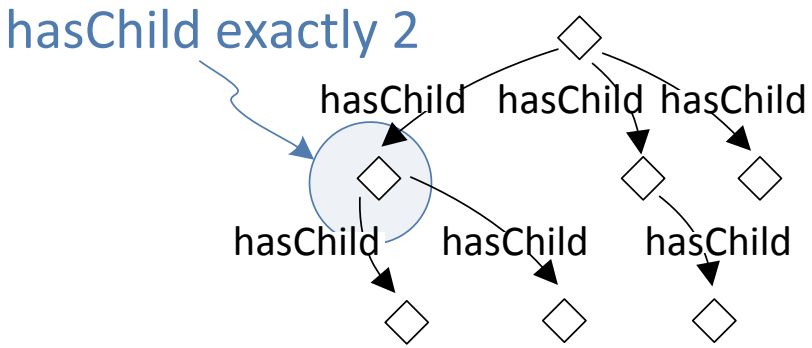
- at least two children"
- at most two children"
- exactly two children"
- at least two children that are parents themselves"



Property Cardinality Restrictions

"Individuals with

- at least two children"
- at most two children"
- exactly two children"
- at least two children that are parents themselves"

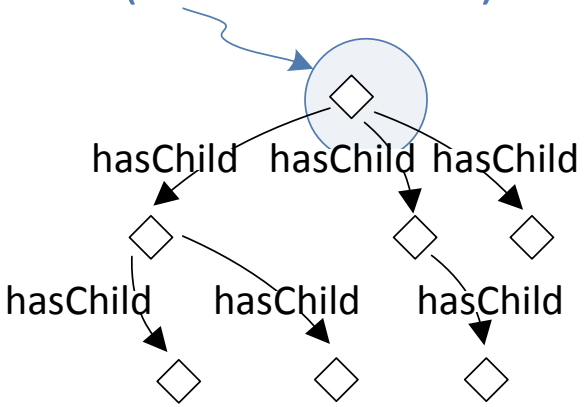


Property Cardinality Restrictions

"Individuals with

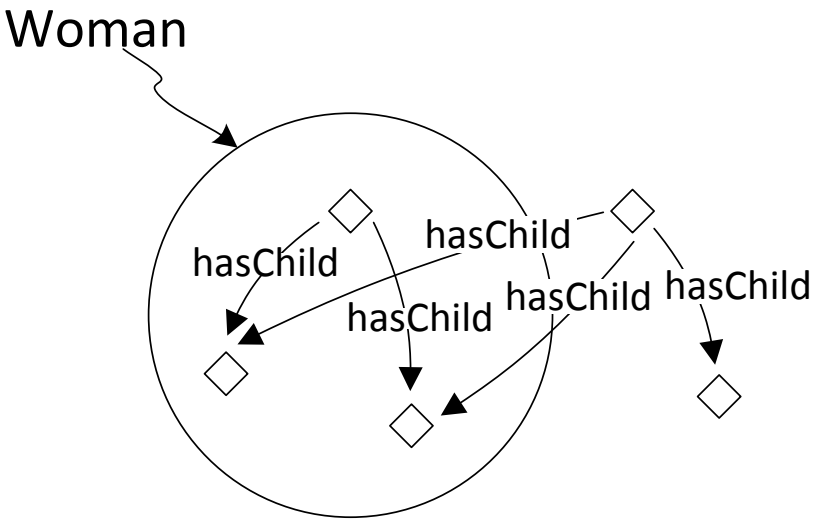
- at least two children"
- at most two children"
- exactly two children"
- at least two children that are parents themselves"

hasChild min 2 (hasChild min 1)



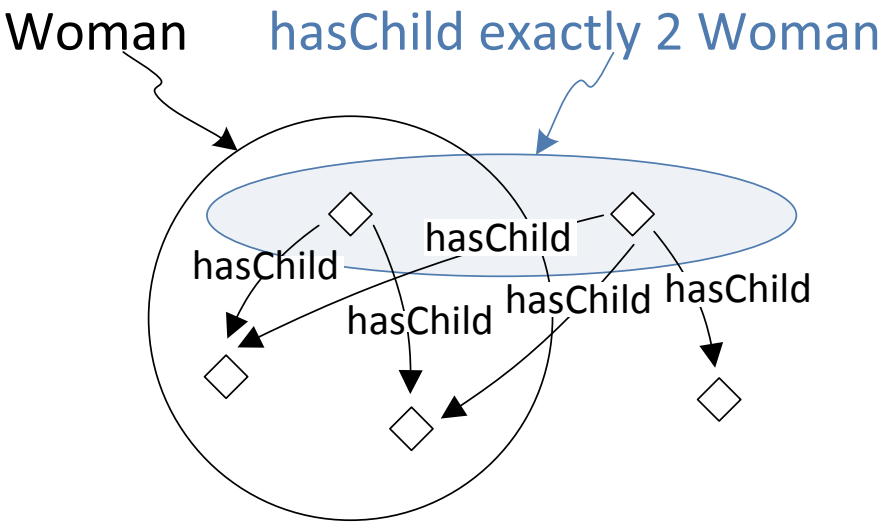
Property Cardinality Restrictions

"Individuals with exactly two daughters"



Property Cardinality Restrictions

"Individuals with exactly two daughters"



Cardinality Restrictions and Identity

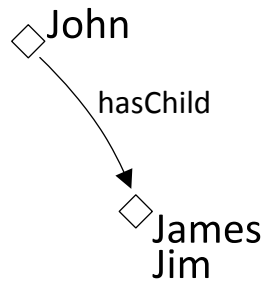
```
Individual: John
   Facts: hasChild James,
          hasChild Jim
```

Can we conclude that John belongs to the following classes?

- hasChild min 2
- hasChild max 2
- hasChild exactly 2
- hasChild min 1

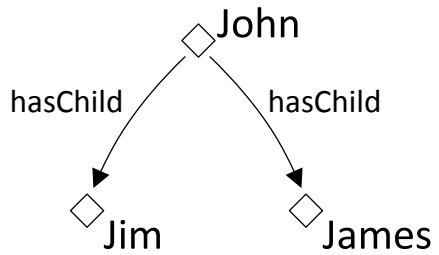
Cardinality Restrictions and Identity

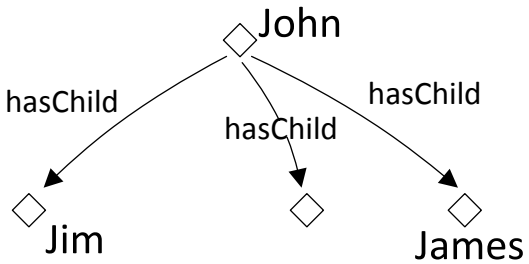
```
Individual: John
   Facts: hasChild James,
          hasChild Jim
```



Can we conclude that John belongs to the following classes?

- hasChild min 2 NO
- hasChild max 2 NO
- hasChild exactly 2 NO
- hasChild min 1 YES



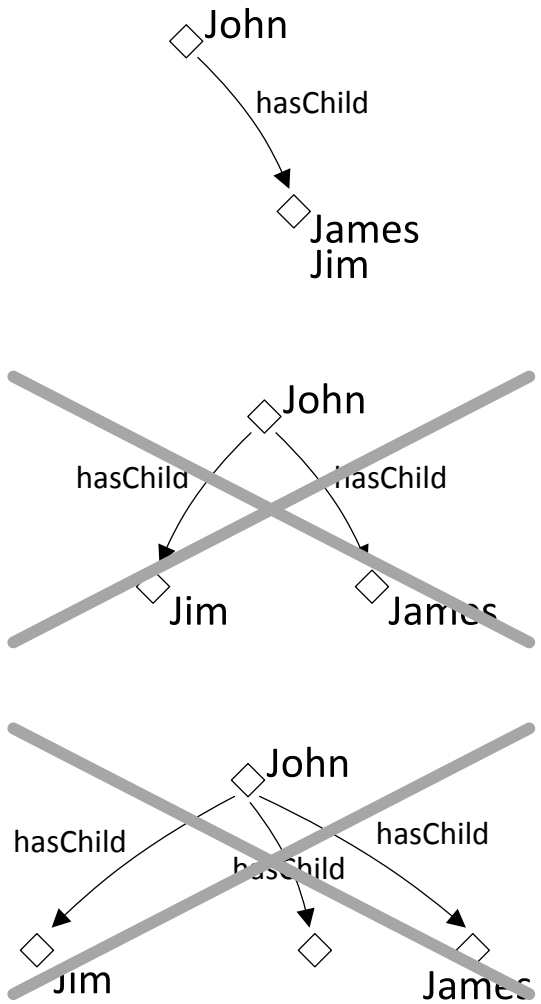


Cardinality Restrictions and Identity

```
Individual: John
   Facts: hasChild James,
          hasChild Jim
   Types: hasChild max 1
```

Can we conclude that John belongs to the following classes?

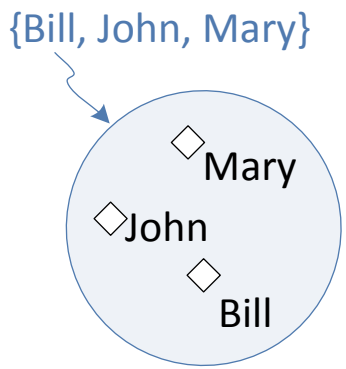
- hasChild min 2 NO
- hasChild max 2 YES
- hasChild exactly 2 NO
- hasChild min 1 YES

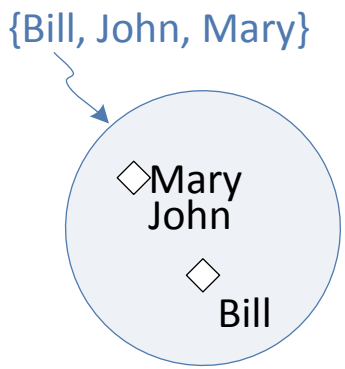


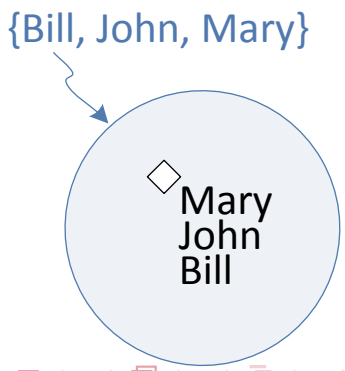
Enumeration of Individuals

```
Class: MyBirthdayGuests
   EquivalentTo: { Bill, John, Mary }
```

(Some) possible interpretations:



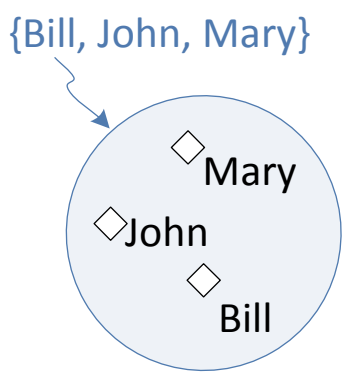


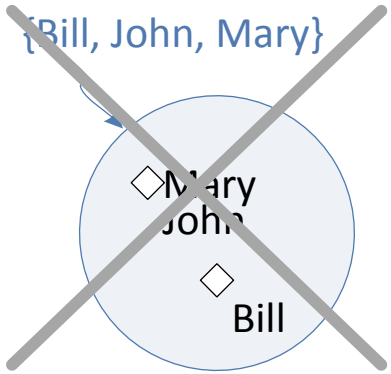


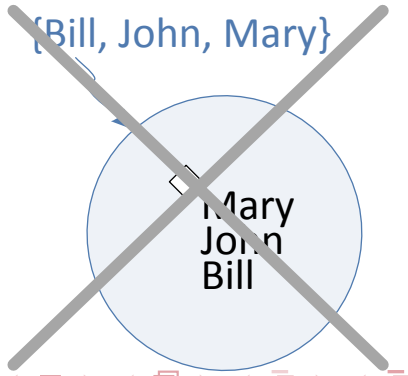
Enumeration of Individuals

```
Class: MyBirthdayGuests
    EquivalentTo: { Bill, John, Mary }

DifferentIndividuals:
    Bill, John, Mary
```

(Some) possible interpretations:



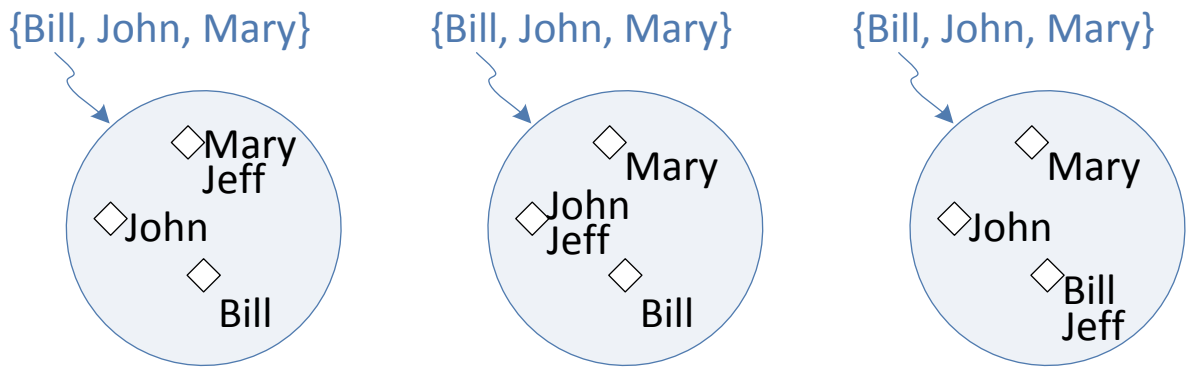




Enumeration of Individuals

```
Class: MyBirthdayGuests
   EquivalentTo: { Bill, John, Mary }

DifferentIndividuals:
   Bill, John, Mary

Individual: Jeff
    Types: MyBirthdayGuests
```

(Some) possible interpretations:



Literatur I

Gruber, T. R. (1995).

Toward principles for the design of ontologies used for knowledge sharing?

Int. J. Hum.-Comput. Stud., 43(5-6):907–928.

Guarino, N., Oberle, D., and Staab, S. (2009).

What is an Ontology?

In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 1–17. Springer.

Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P. F., and Rudolph, S.

Owl 2 web ontology language primer : W3c recommendation 27 october 2009.

Literatur II

Maedche, A., Staab, S., and Studer, R. (2001).
Ontologien.
Wirtschaftsinformatik, 43(4):393–396.

Studer, R., Benjamins, V. R., and Fensel, D. (1998).
Knowledge engineering: Principles and methods.
Data Knowl. Eng., 25(1-2):161–197.