3rd International Symposium ISKO-Maghreb'2013
Concepts and Tools for Knowledge Management (KM)
Marrakech (Morocco): November 8th. and 9th., 2013

Extracted fom:
Competence Discovery and Composition

Badrina GASMI BOUMEZOUED: University of Béjaïa/ReSyD, Algeria

Hassina NACER: University of Béjaïa/ReSyD, Algeria

Nacer BOUDJLIDA: University of Lorraine/LORIA, France

Introduction

Proposal

- Competence Representation: Graph-Based (Conceptual Graphs)
- Comptence Discovery and Composition: Reasonning based on operations on the graphs

Presentation Outline

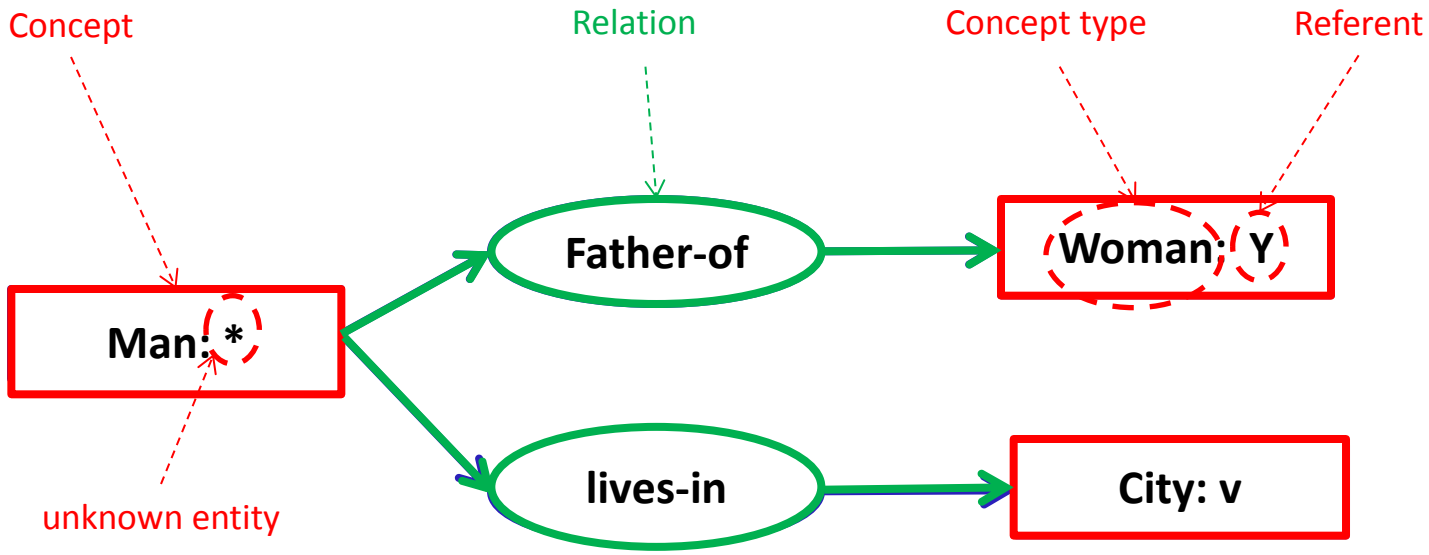
1. Introduction
2. Introduction to Conceptual graphs
3. Competence representation
4. Competence discovery and composition
5. Implementation
6. Conclusion

Conceptuel graphs

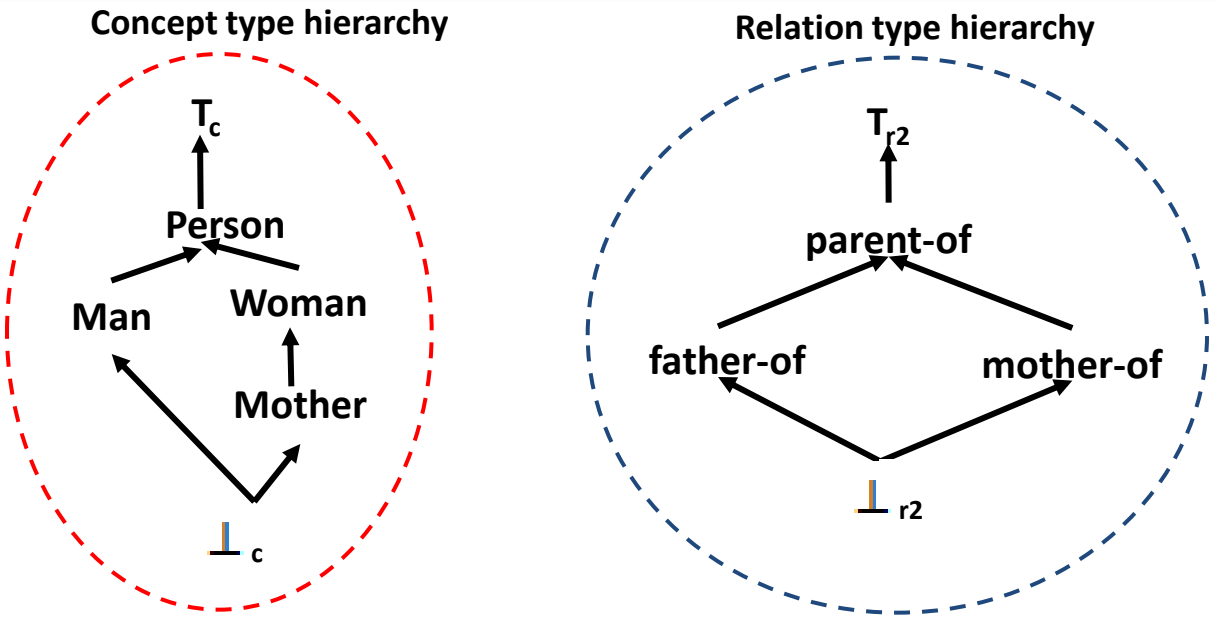
Graph:

- Bi-parted
- Oriented
- Labelled
- Finite
- Not necessarily connected

Conceptual graph example



Conceptual Graphs Support



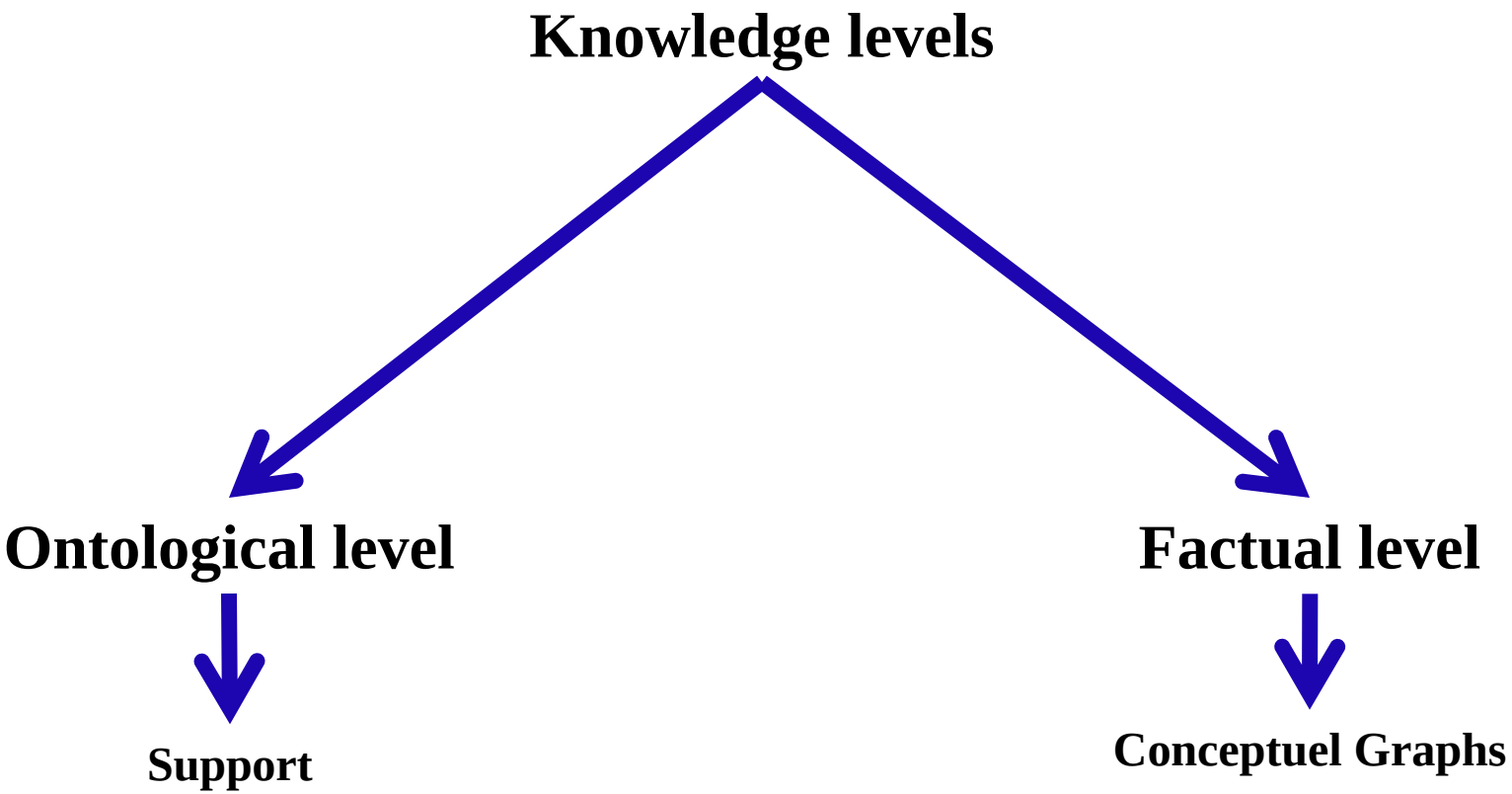
The set of referents (I):
I={man1, man2, woman1, mother1}

The conformity function (τ):
τ(man1)=Man, τ(man2)=Man, τ(woman1)=Woman, τ(mother1)=Mother.

The relation signatures:
Tr2(Tc, Tc), parent-of(Person, Person), mother-of(woman, Person), father-of(Man, Person),
$\perp_{r2}$ $\perp_c$, $\perp_c$)

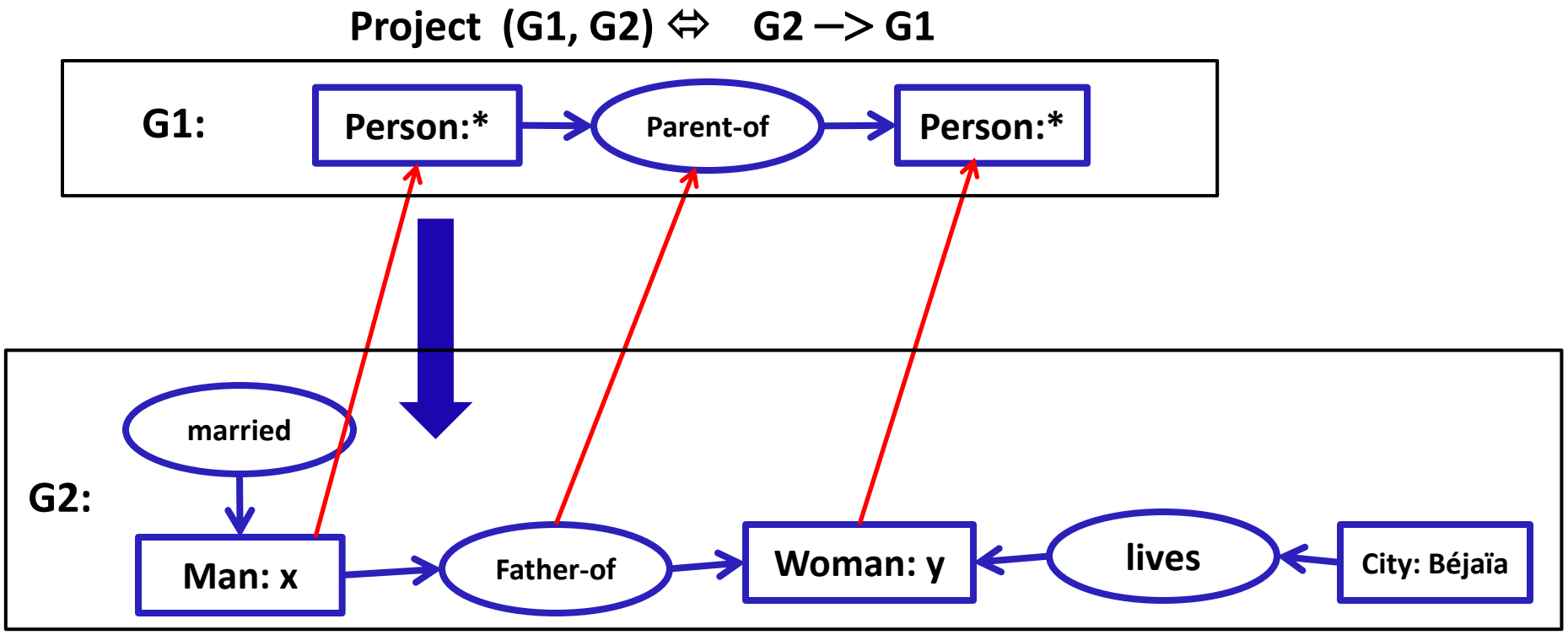
Conceptuel Graphs

Knowledge Description Types



Operating on Conceptual Graphs

- Projection: graph homomorphism thanks to the projection of
  1. Concept nodes on Concept nodes.
  2. Relation nodes on Relation nodes

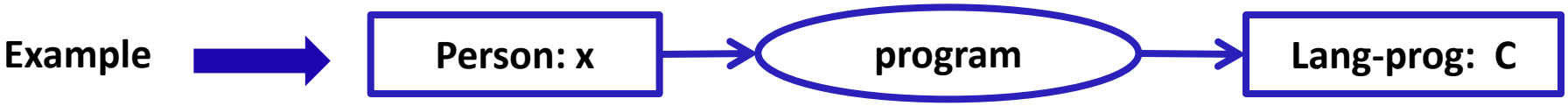


Presentation Outline

1. Introduction
2. Introduction to Conceptual graph formalism
3. Competence representation
4. Competence discovery and composition
5. Implementation
6. Conclusion

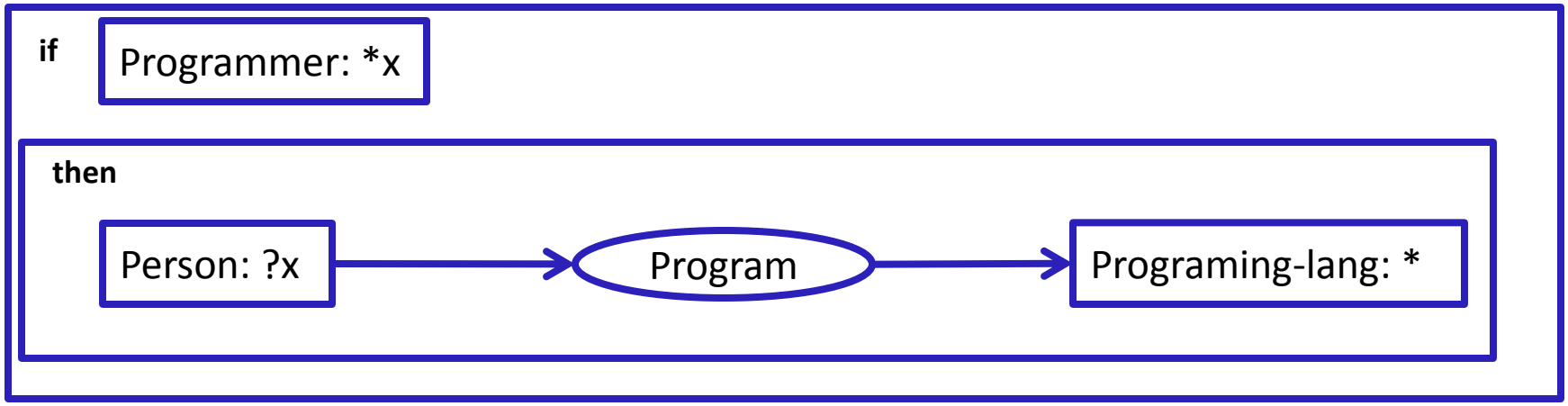
CG-based Competence Representation

- Competence exporter ---> Concept
- Competence ---> Relation
- Generic information and properties ---> Graph rules

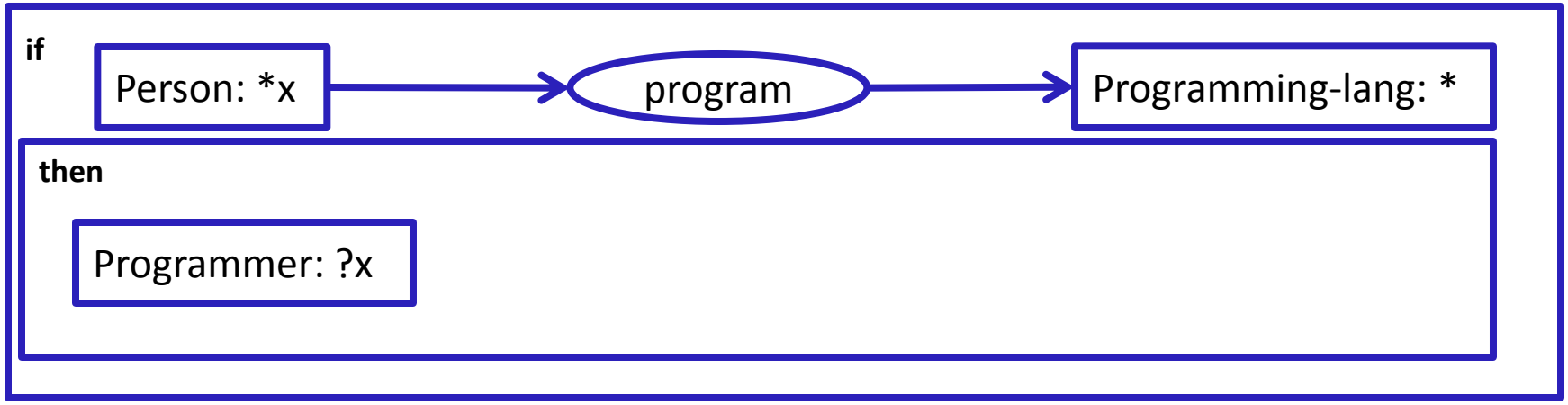


Conceptual graph rule Example

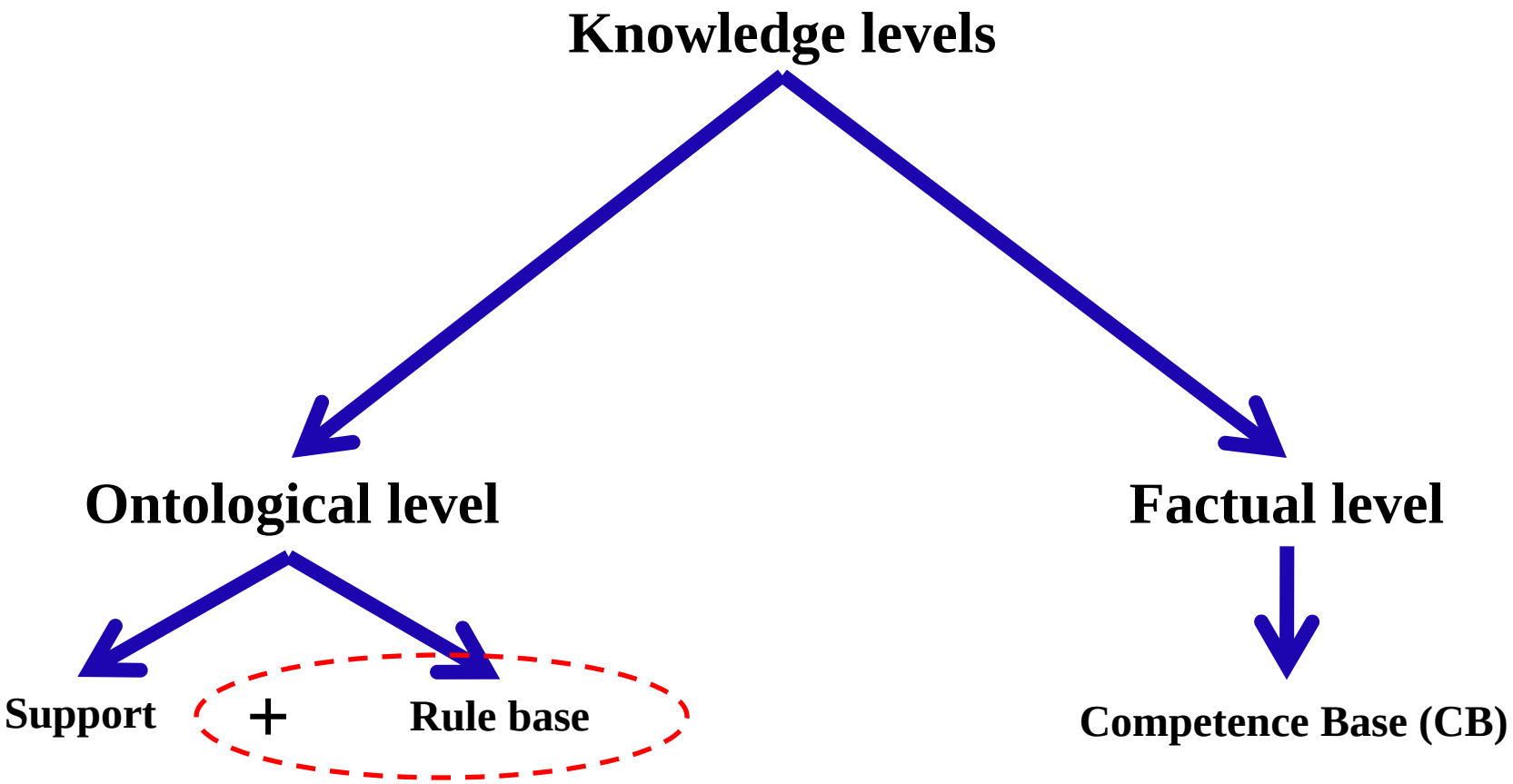
R1:



R2:



Knowledge levels

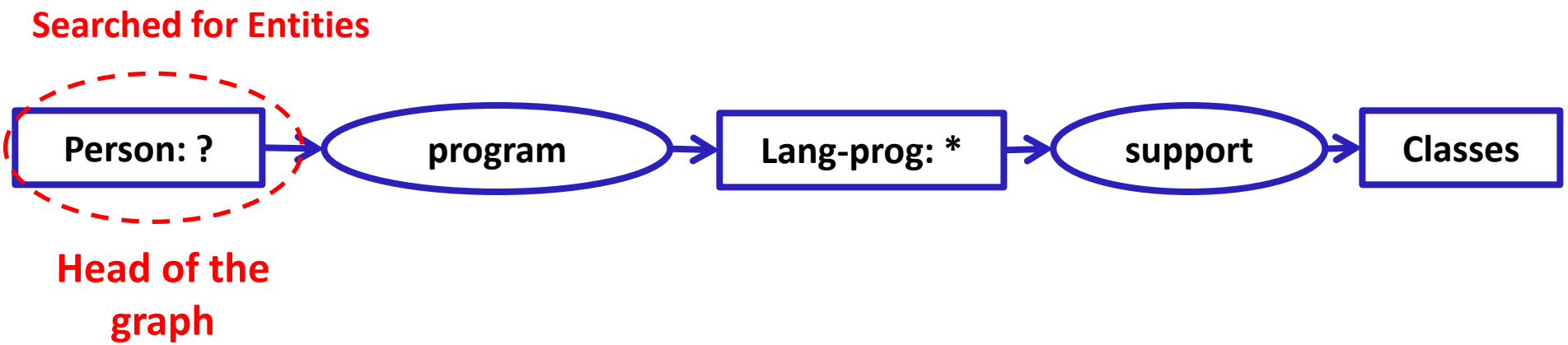


Plan

1. Introduction
2. Introduction to Conceptual graph formalism
3. Competence representation
4. Competence discovery and composition
5. Implementation
6. Conclusion

Query Representation and Example

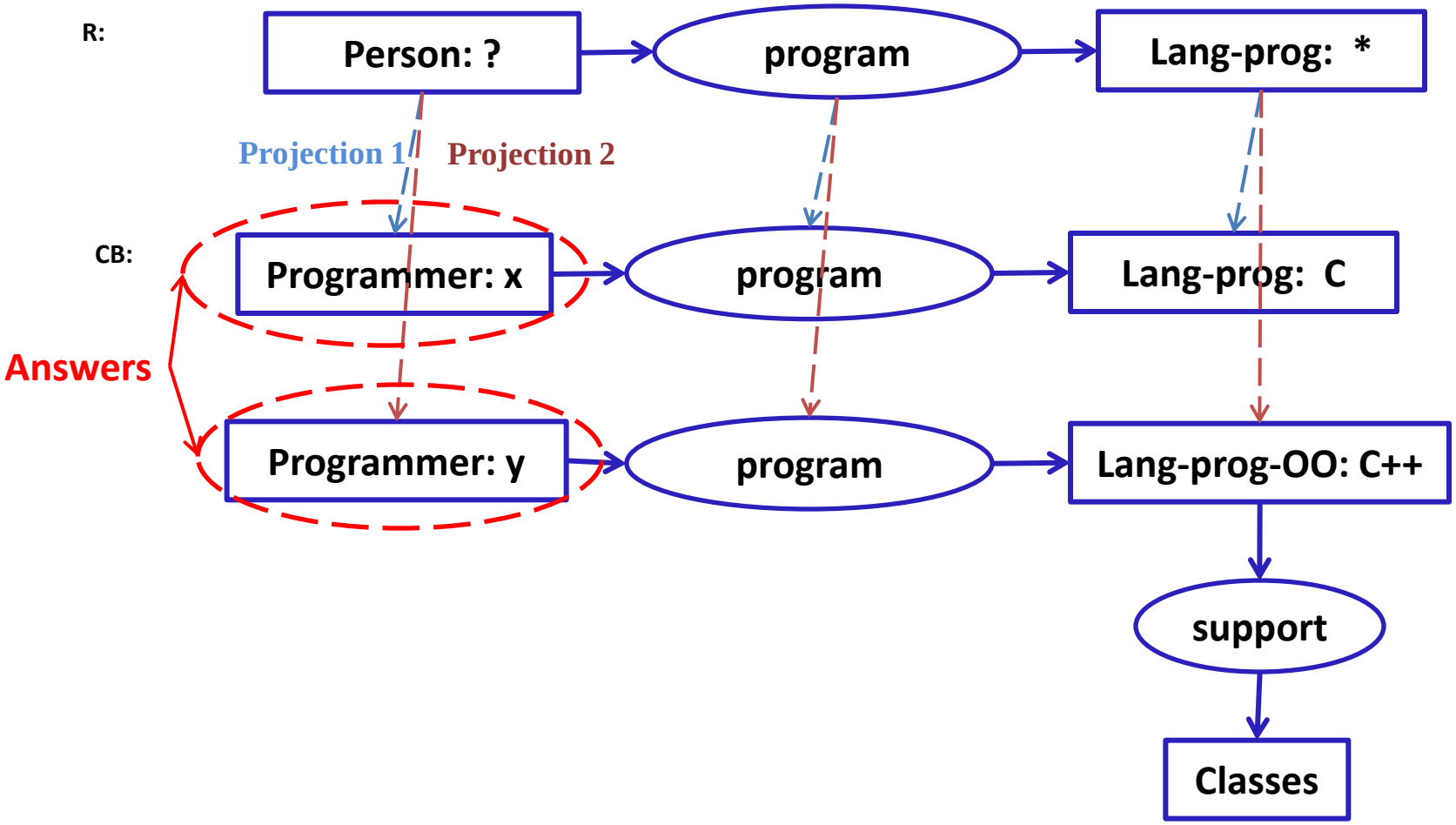
- ***Query Q ---> Headed graph***



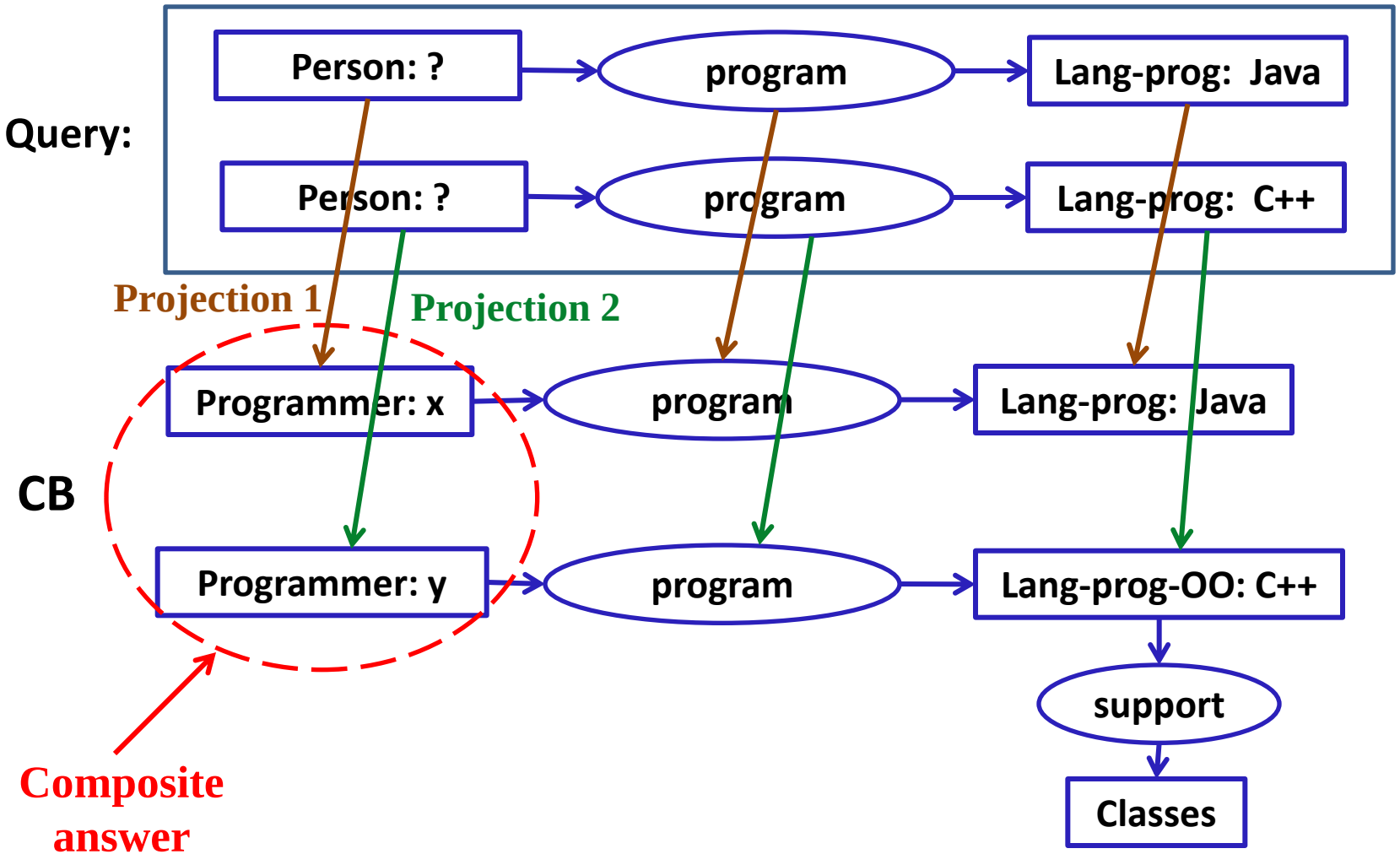
- ***Query Evaluation***

1. Project the Query-Graph Q on the Competence Base (CB)
2. Select the projections of the head node of Q

Example: Query Evaluation and Exact satisfaction

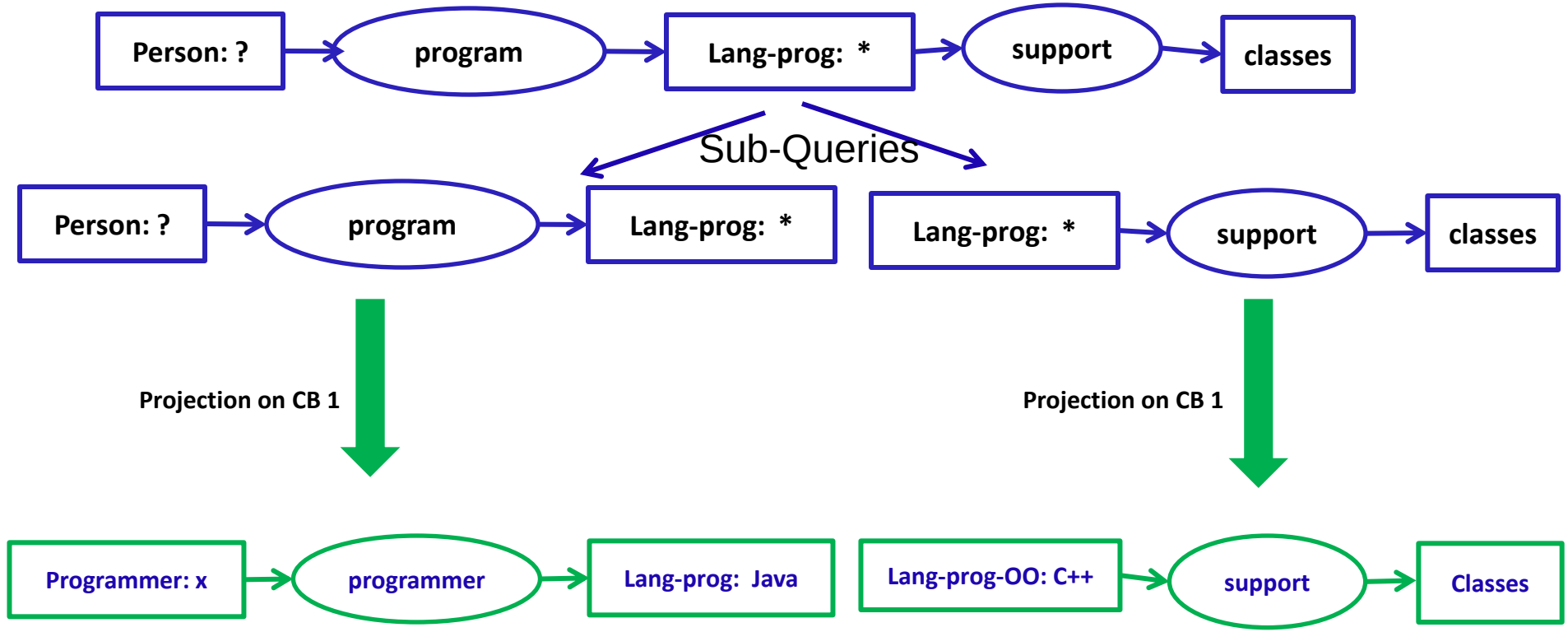


Composite Answer Example

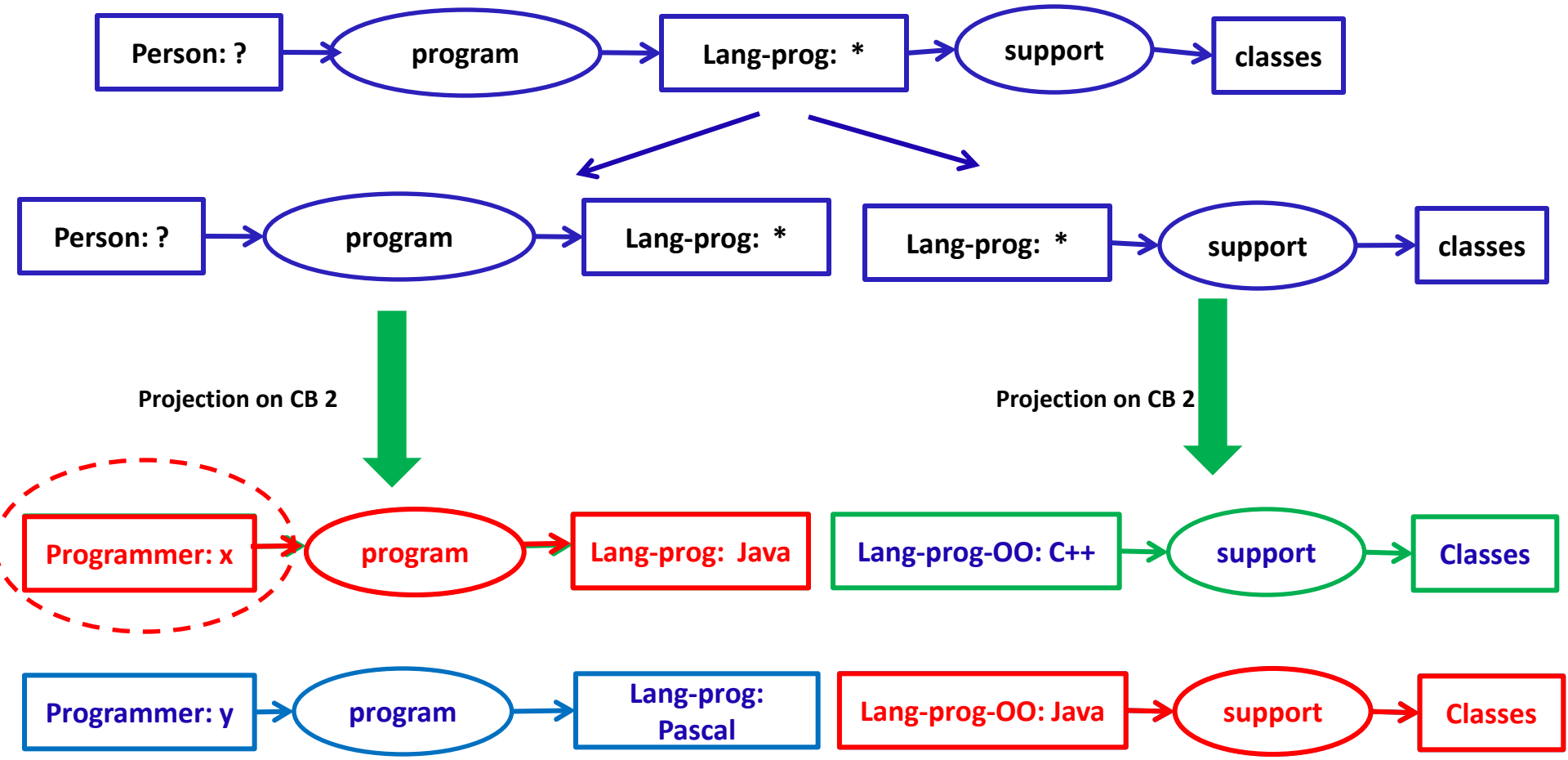


Distributed Satisfaction Example

Mediator 1



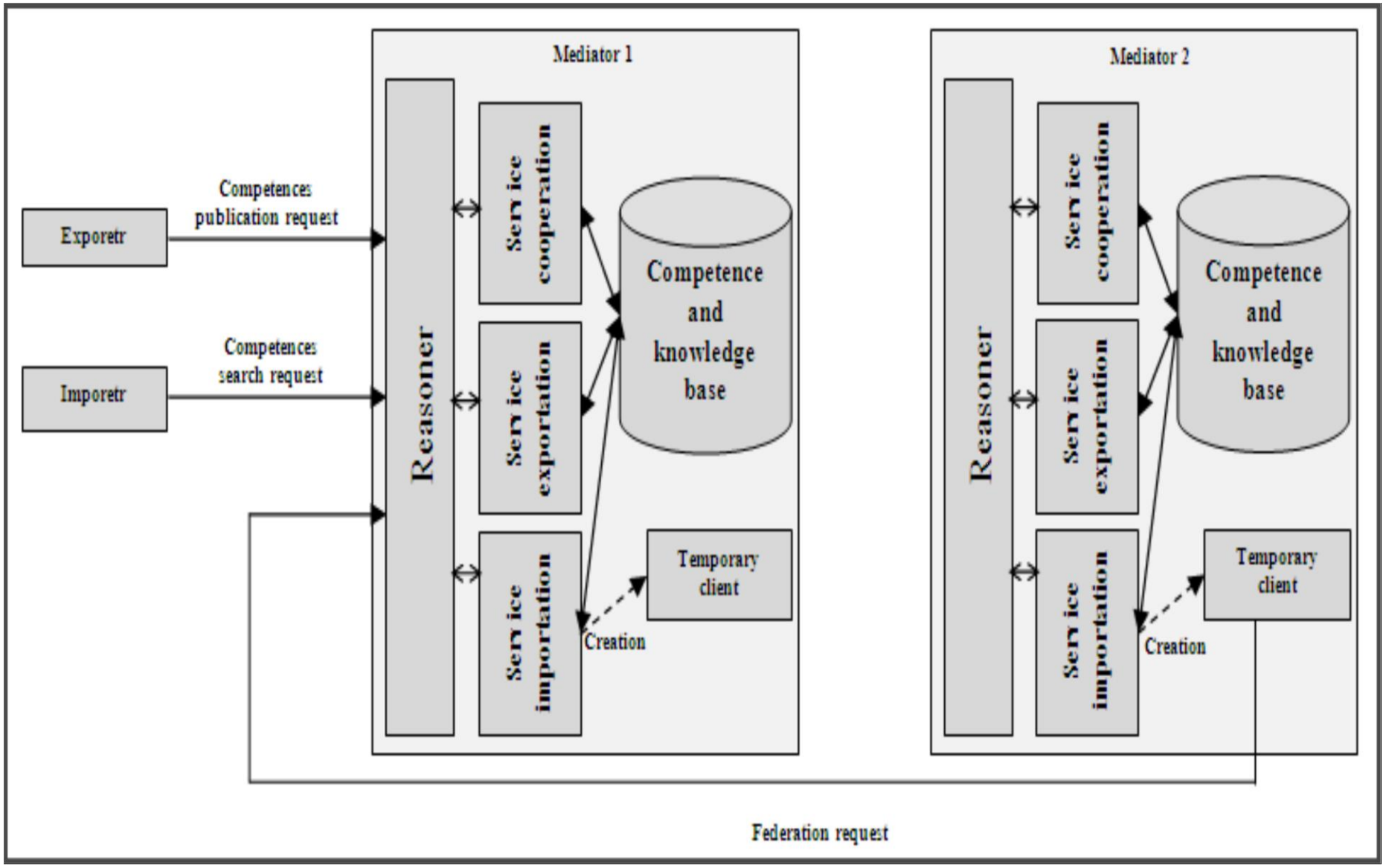
Mediator 2


Person: ?
program
Lang-prog: *
support
classes
Person: ?
program
Lang-prog: *
Lang-prog: *
support
classes
Projection on CB 2
Projection on CB 2
Programmer: x
program
Lang-prog: Java
Lang-prog-OO: C++
support
Classes
Programmer: y
program
Lang-prog: Pascal
Lang-prog-OO: Java
support
Classes

Plan

1. Introduction
2. Introduction to Conceptual graph formalism
3. Competence representation
4. Competence discovery and composition
5. Implementation
6. Conclusion

The Prototype's Architecture



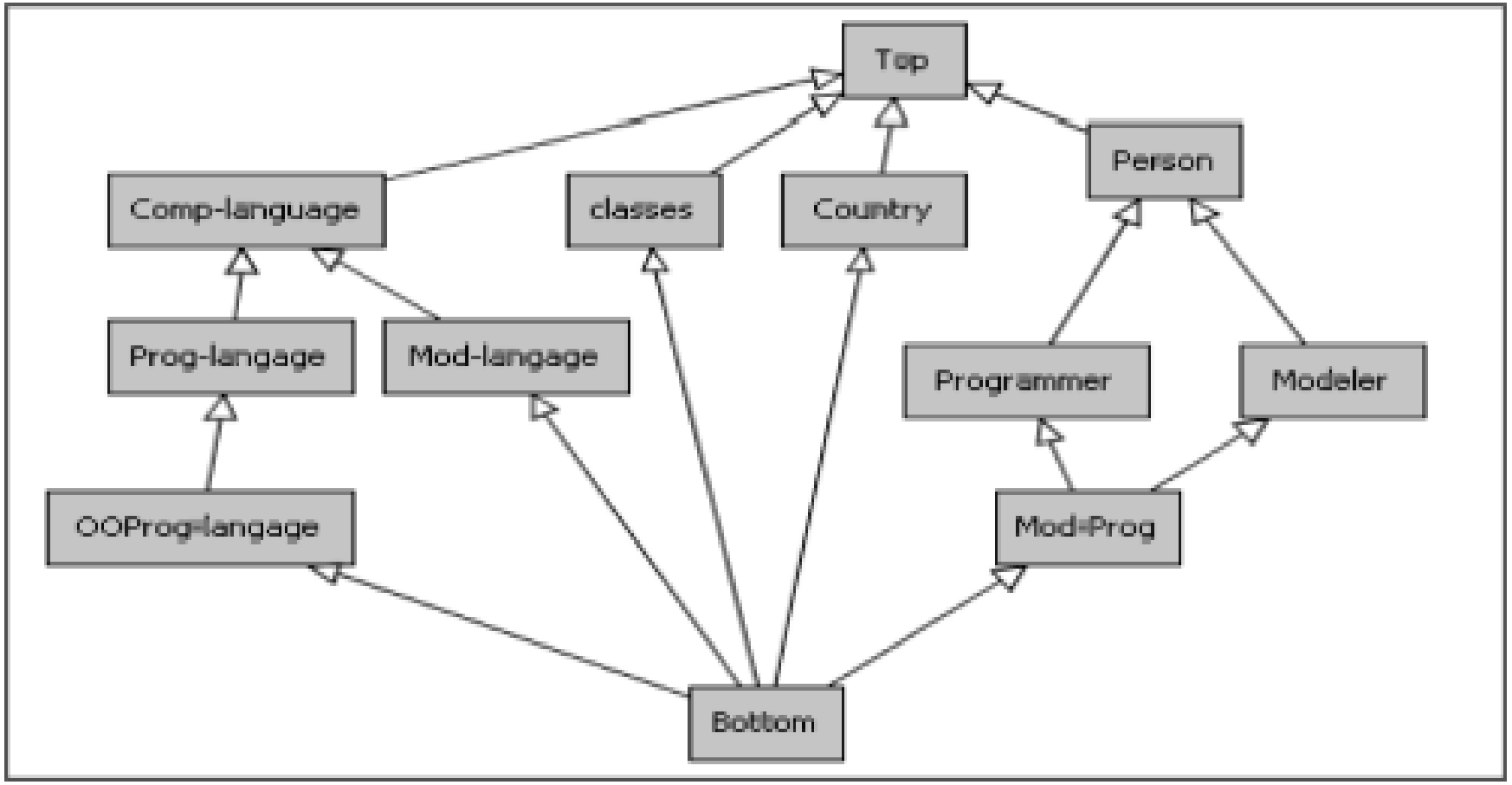
The Prototype's Architecture

- Based on CoGITaNT's Client/Server Architecture and its Library
- **CoGITaNT: C**onceptual **G**raphs **I**ntegrated **T**ools allowing **N**ested **T**yped graphs
- Open source, LIRM Montpellier, CNRS, France
- Library of C++ classes which allows developing applications based on the CG knowledge representation scheme
- Multiple presentations of CGs: linear, Graphical, CGIF, BCGCT (proper form), CoGXML (XML).

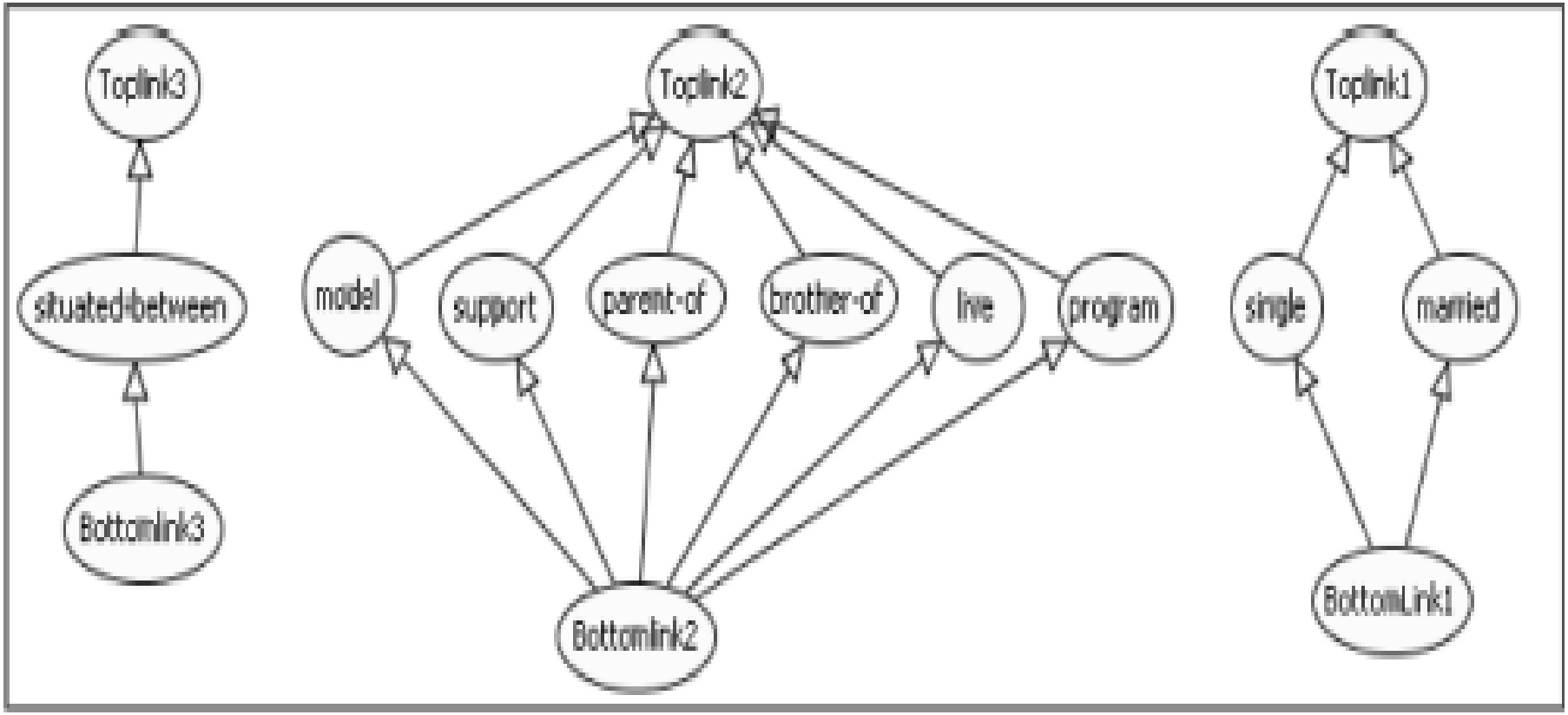
The Prototype's Architecture

- Based on the client/server architecture of CoGITaNT
- CoGITaNT's **clients** represent either a **competence exporter** or a **competence importer**
- CoGITaNT's **servers** represent **mediators**
- **Temporary Clients** :
  - Created by CoGITanT
  - In order to possibly cooperate with other mediators

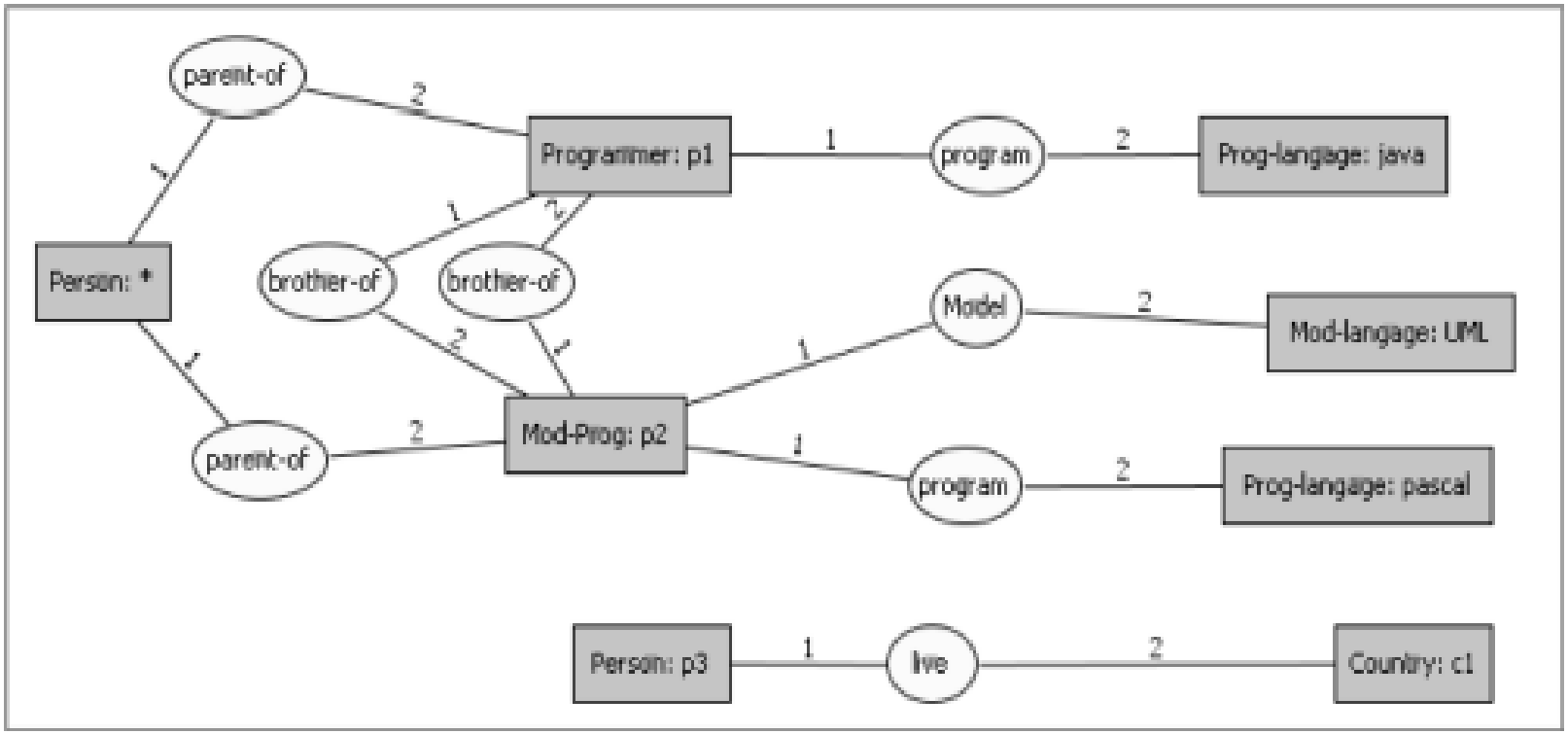
The Hierarchy of concepts



The Hierarchy of Relation Types

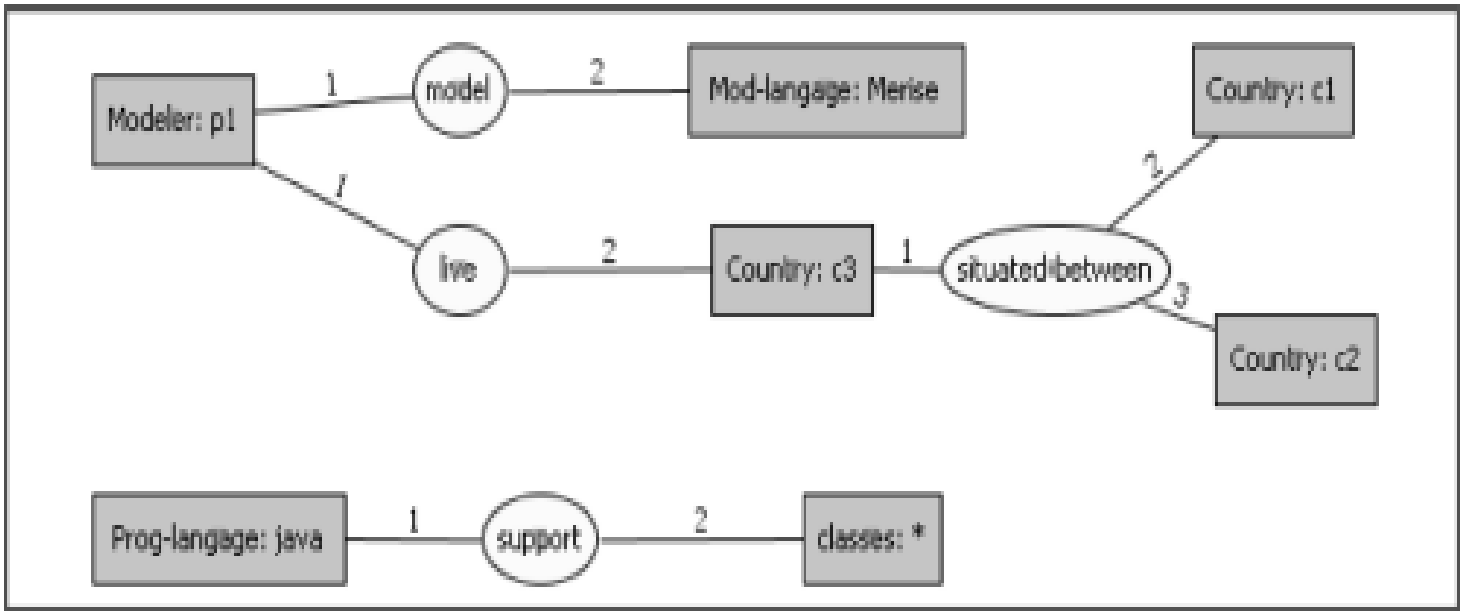


Competence base of the mediator M1



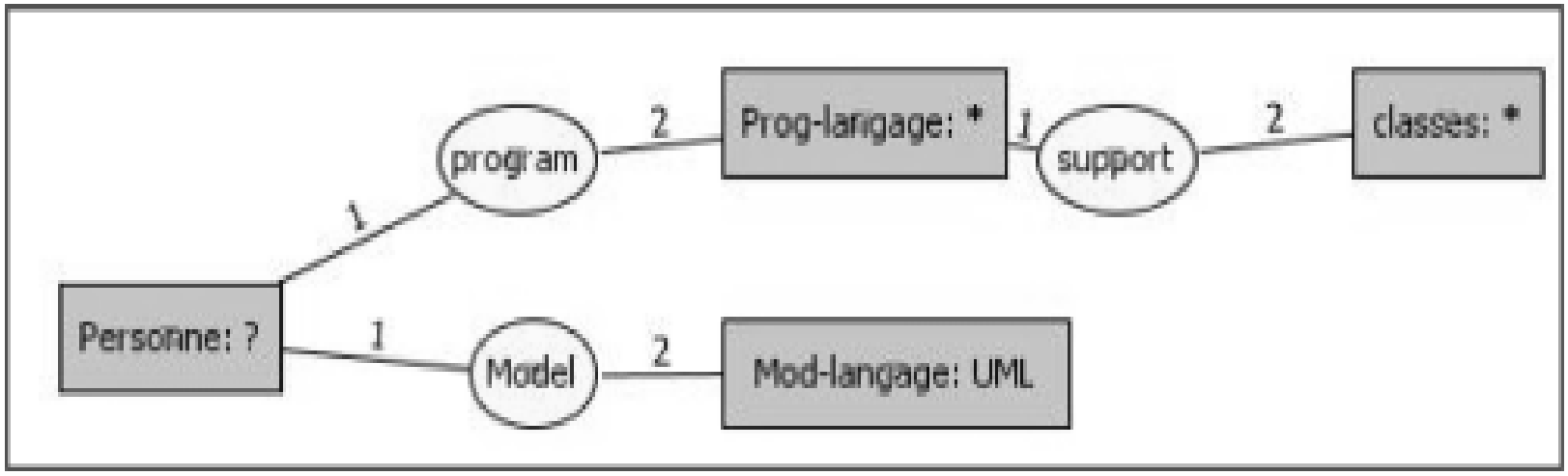
Execution scenario

Competence base of the mediator M2



Execution scenario

The Request



Execution scenario

Mediator M1

```
===>>>Mediator's adress is 192.168.0.1

 ------------------------------------------------------------------------
_graph1:
       [classes]<-(support)<-[Prog-langage]<-(program)<-[Person:?]->(model)->[Mod-langage:UML].

 ------------------------------------------------------------------------
===>>>This request is not satisfied locally

 ------------------------------------------------------------------------
===>>>Searching composite answers

 ------------------------------------------------------------------------
===>>>Sub-request 1:_graph2:
       [Person:?]->(model)->[Mod-langage:UML].
=====>>>Satisfied. Answer is: [Mod-Prog:p2]

   >>Sub-request 2:_graph9:
       [Person:?]->(program)->[Prog-langage]->(support)->[classes].
    >>Not satisfied locally

 ------------------------------------------------------------------------
=====>>>Distributed satisfaction:
 ------------------------------------------------------------------------
   >>Partial satisfactions are:
     >>1. _graph10:
       [Mod-Prog:p2]->(program)->[Prog-langage:pascal].
     >>2. _graph11:
       [Programmer:p1]->(program)->[Prog-langage:java].

 ------------------------------------------------------------------------
====>>>Sending partial satisfactions to the mediator having the adress 192.168.0.2

 ------------------------------------------------------------------------
====>>>Receiving answers      >>[Programmer: p1]
 ------------------------------------------------------------------------
====>>>There is a composite answer to the request: ([Mod-Prog: p2],[Programmer: p1])
```

Mediator M2

```
 ------------------------------------------------------------------------------
===>>>Mediator's adress is 192.168.0.2

 ------------------------------------------------------------------------------
===>>>The received request is:_graph9:
        [Person:?]->(program)->[Prog-langage]->(support)->[classes].
     >>Not satisfied locally

 ------------------------------------------------------------------------------
====>>>Received partial satisfactions:
      >>1. _graph10:
        [Mod-Prog:p2]->(program)->[Prog-langage:pascal].
      >>2. _graph11:
        [Programmer:p1]->(program)->[Prog-langage:java].

 ------------------------------------------------------------------------------
=====>>>Localpartial satisfactions:
      >>1. _graph12:
        [Prog-langage:java]->(support)->[classes].

 ------------------------------------------------------------------------------
======>>>Satisfied. Answer is: [Programmer:p1]
 =====>>>Sending the answer to the mediator having the adress 192.168.0.1
```

Presentation Outline

1. Introduction
2. Introduction to Conceptual graph formalism
3. Competence representation
4. Competence discovery and composition
5. Implementation
6. Concluding Remarks

Concluding Remarks

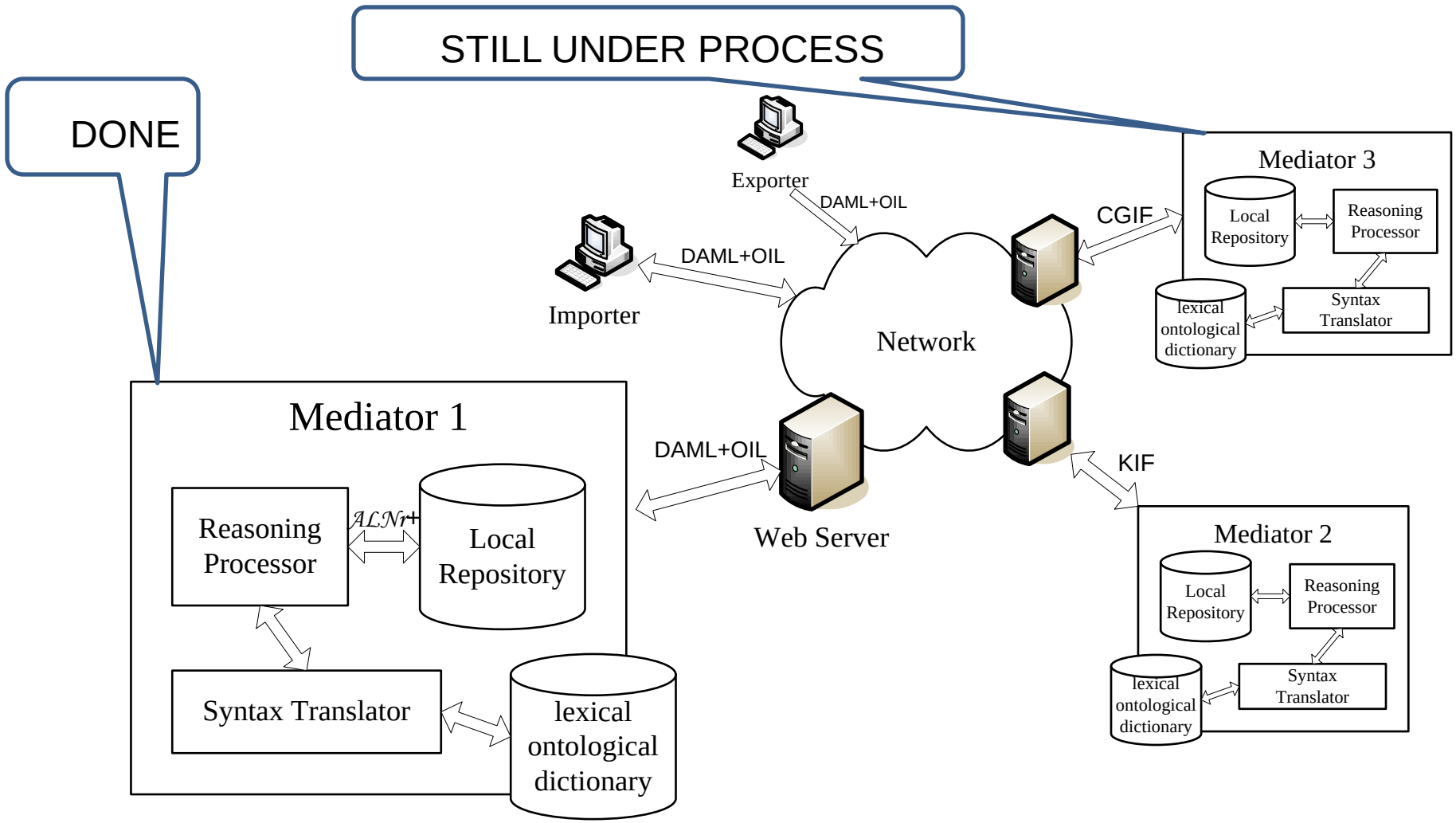
Why conceptual graphs?

- Generality: other formalisms (like DL) are easily expressible as CGs
- Power to express n-ary relationships
- Simplicity of the model and facility of its understanding
- Readability of knowledge as well as reasonning.
- No necessity to learn a logical formalism in order to do logical reasonning.
- Existence of standards (CGIF)
- Existence of tools implementing operations on graphs

Concluding Remarks

- CG Main Drawback: Projection complexity
- NP-Complete
- Polynomial algorithms do exist (for particular CGs, like trees)
- Complexity in M^N
  - M: number of nodes in the query-graph
  - N: number of nodes of the graph on which the projection is operated
- In this work: Polynomial too
- Algorithm complexity = a matter of **further work**
- An other matter of **further work**: dynamic and semantic discovery of cooperating mediators

Concluding Remarks: Long Term Objective



That's all Folks!

Thank You For Paying Attention

Any Question?