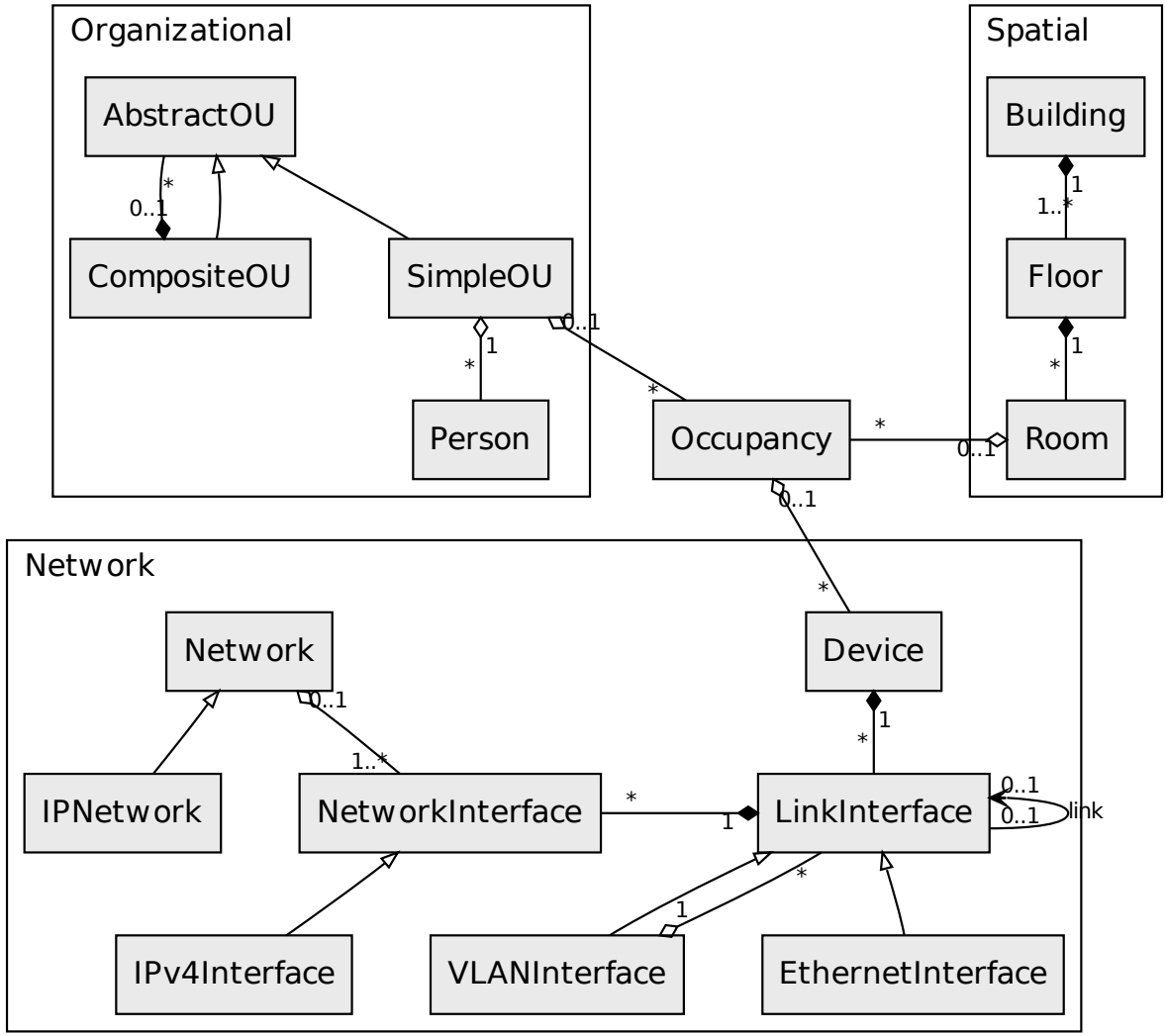
Laconic Object Query Language Using Features of Object Model

V. Dimitrov

Petrozavodsk State University
Department of Computer Science

Object model SON



Example queries on SON (JP-QL)

- **Find router for specified device:**

```
select d from Device as d
        left join d.linkInterfaces as lis
        left join lis.networkInterfaces as nis
        left join nis.network.networkInterfaces as nis2
where nis2.linkInterface.device.forwarding = true and d.id = 25
```

- **Find devices into main building:**

```
select d from Device as d
        left join d.occupancies as os
where os.room.floor.building.name = "Main Building"
```

Example queries on SON (Criteria Query 2.0)

- **Find devices into main building:**

```
CriteriaBuilder builder = em.getCriteriaBuilder();
CriteriaQuery<Tuple> cr = builder.createTupleQuery();

Root<Device> root = cr.from(Device.class);
Join<?, ?> join = root.join("occupancy").join("room");
join = join.join("floor").join("building");
cr.where(builder.equal(join.get("name"), "TK"));
cr.multiselect(root.get("id"));
```

Tasks and requirements

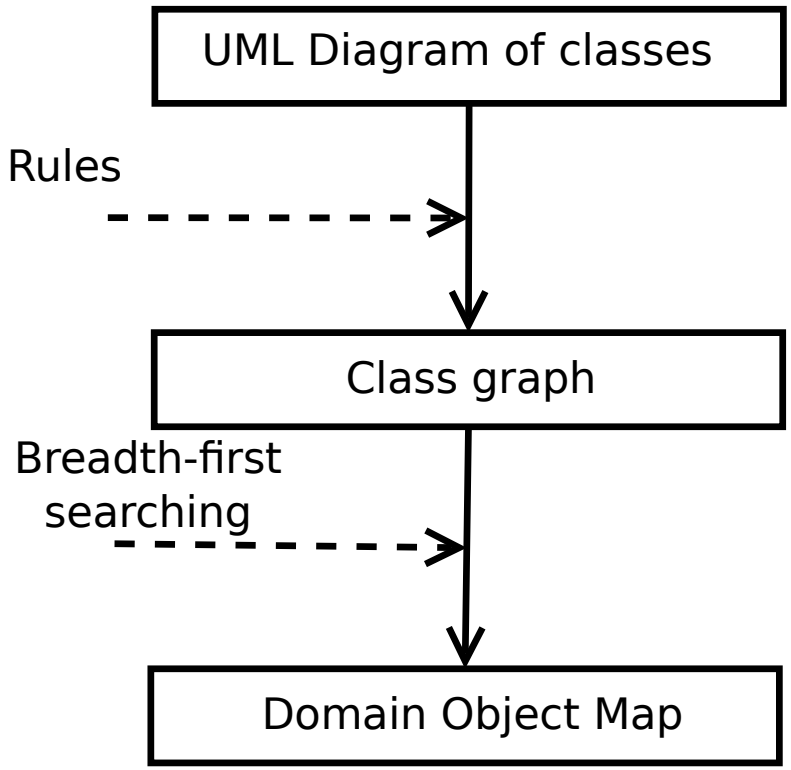
- Simple tool for working with domain (search, select and so on).
- Independence from storage (SQL DB, no-SQL DB, object DB, file system storage and so on).
- Uniform access to storage and extra source of information.
- The tool should provide fast generation of large amount of different types of queries.
- Laconic text of query.

Problems

- How to laconic query text?
- Description of domain.
- Path-expression.
- Problems of links.
- Performance of tool.

Main idea

- Creating base map with paths due to breadth-first search algorithm into class graph for using into object graph.
- Manual changing DOM due to domain expert opinion.

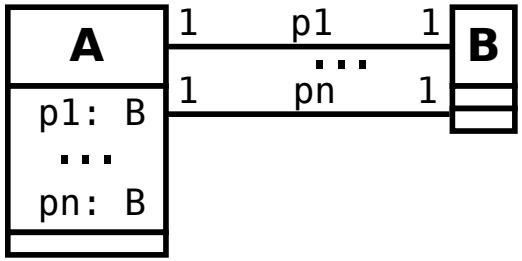


Using links from class diagram for searching in object graph

Links	LaOQL support
Association	
types:	
simple	DOM
composition	DOM
aggregation	DOM
class association	DOM
attributes:	
direction	DOM
multiplicity	DOM,runtime
Generalizaion	
types:	
single	DOM, runtime
multiple	DOM
Implementation	runtime
Dependency	-

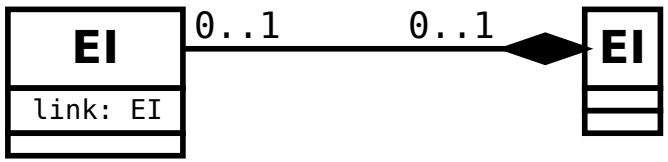
1 (association)

if class A has association link with class B throught not empty set attributes P, then path is transitions throught attributes from P.



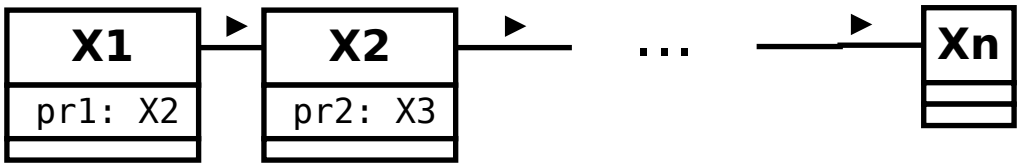
$\mathrm{P} = \{\mathrm{p_1}, \ldots, \mathrm{p_n}\}$
Paths: $[[\mathrm{p_1}], \ldots, [\mathrm{p_n}]]$
DOM: $\{\mathrm{A}\ \{\mathrm{B}\ [[\mathrm{p_1}], \ldots, [\mathrm{p_n}]]\}\}$

Example from SON: EthernetInterface$\rightarrow$EthernetInterface
DOM: $\{\mathrm{EthernetInterface}\ \{\mathrm{EthernetInterface}\ [[\mathrm{link}]]\}\}$



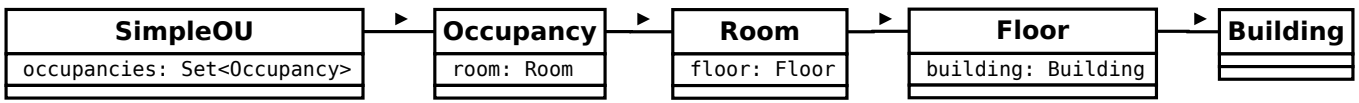
2 (set of associations)

If class $\mathrm{X_1}$ has link to class $\mathrm{X_n}$ throught set of classes $\mathrm{X_2}, \ldots, \mathrm{X_{n-1}}$ due to attributes $\mathrm{pr_1}, \ldots, \mathrm{pr_{n-1}}$ respectively, then path is transition throught attributes $\mathrm{pr_1}, \ldots, \mathrm{pr_{n-1}}$.



Path: $[[\mathrm{pr_1}, \ldots, \mathrm{pr_{n-1}}]]$
DOM: {A {B $[[\mathrm{pr_1}, \ldots, \mathrm{pr_{n-1}}]]$}}

Example from SON: SimpleOU→Building
DOM: {SimpleOU {Building [[occupancies room floor building]]}}

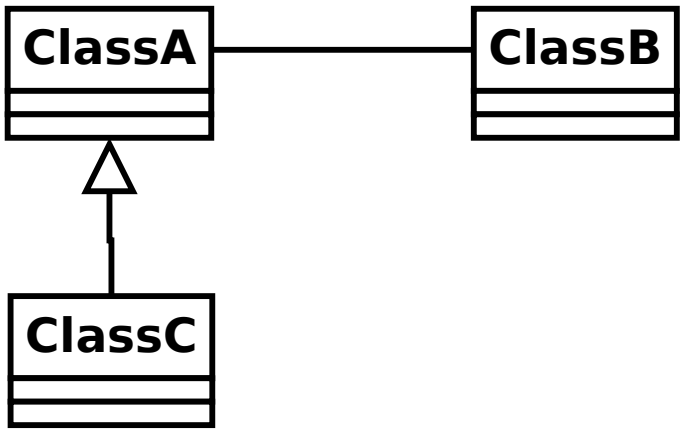


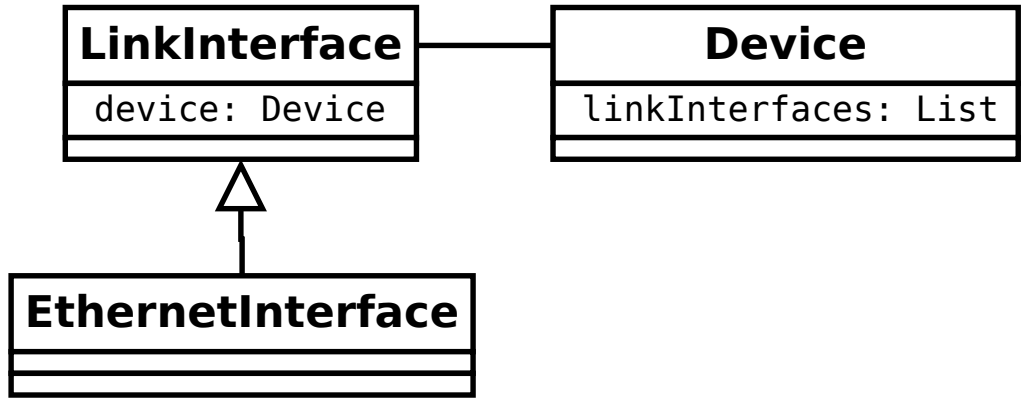
3 (generalization and association with parent)

If class C extends from class A and A has association link with B and C and B have not direct association link then path from $\mathrm{C} \rightarrow \mathrm{B}$ (or $\mathrm{B} \rightarrow \mathrm{C}$) is path $\mathrm{A} \rightarrow \mathrm{B}$ ($\mathrm{B} \rightarrow \mathrm{A}$).

Example from SON: EthernetInterface$\rightarrow$ Device и Device $\rightarrow$ EthernetInterface

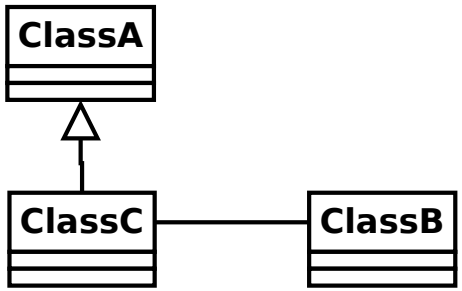
DOM: {EthernetInterface {Device [[“device”]]}
Device {EthernetInterface [[“linkInterfaces”]]}
}





4 (generalization and association with children)

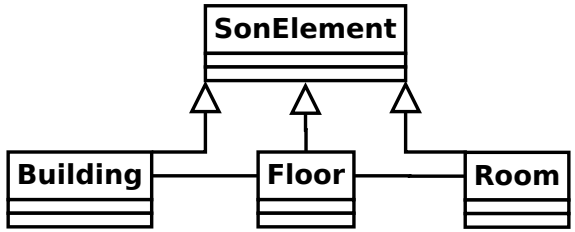
If class C extends from A and class C has association with class B and A and B have not direct association link, then if class A is class C, then path from $\mathrm{C} \rightarrow \mathrm{A}$ is path $\mathrm{C} \rightarrow \mathrm{B}$.



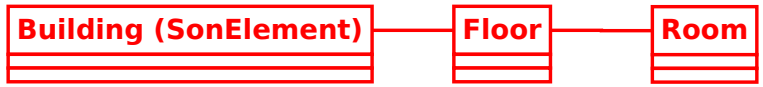
Note: this rule executes in runtime.

4 (example from SON)

sonelement (room)



sonelement — is building

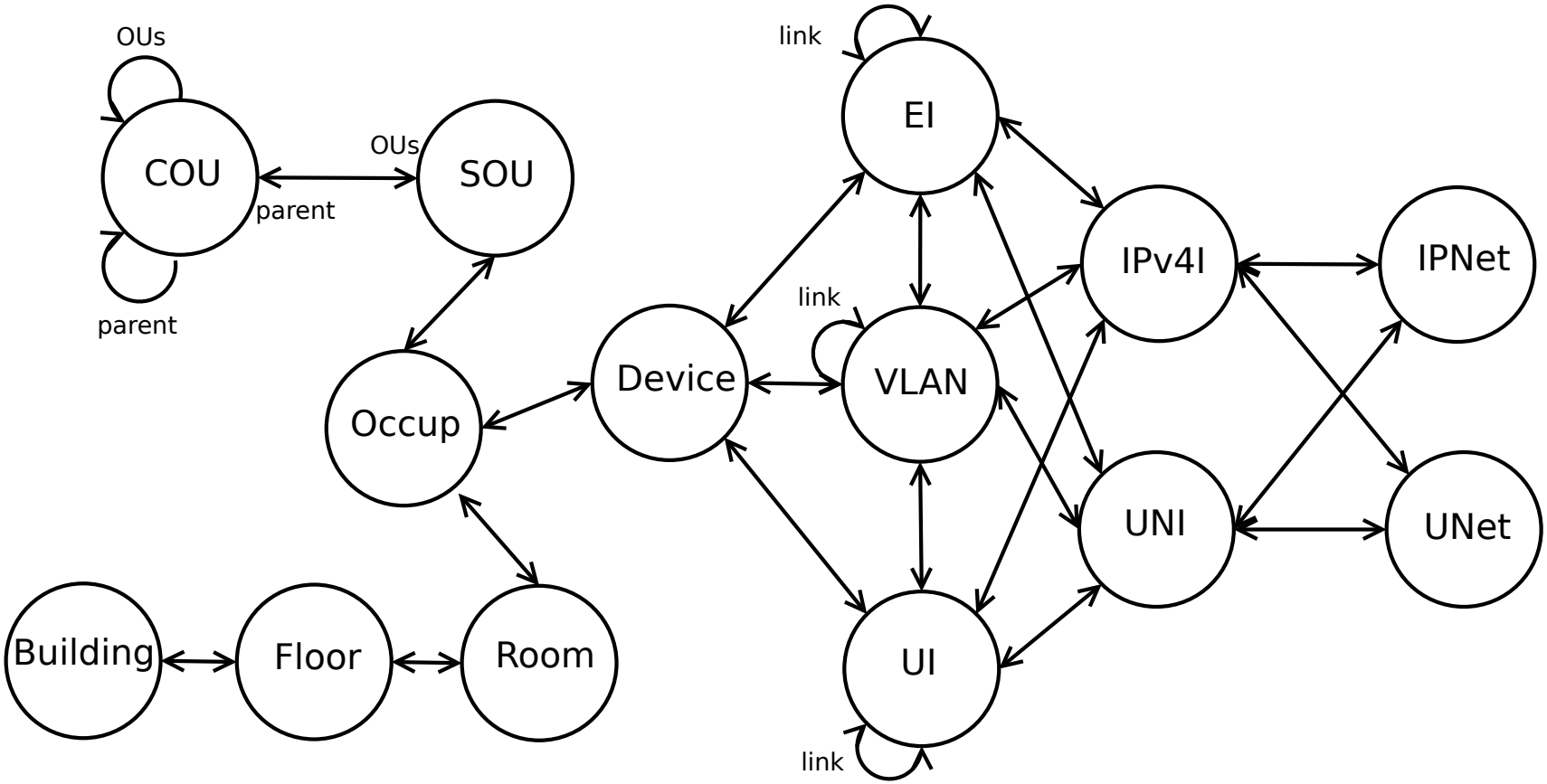


sonelement — is floor

Order:

- define class;
- get path from DOM for this class.

Class graph of SON



Algorithm (main function)

Algorithm 1 Algorithm for creating DOM

```
1: CLASSES — list of classes
2: DOM — map
3: for all source ∈ CLASSES do
4:     for all target ∈ CLASSES do
5:         paths ← (get-paths source target)
6:         DOM ← (assoc-in DOM [source target] paths)
7:     end for
8: end for
```

Path-expression

Our version of path-expression

- Path from class to class by associations links:

  clazz1.clazz2

  clazz1 and *clazz2* have association link

- Eliminate intermediate classes in paths:

  clazz1.clazz3

  clazz1 and *clazz3* have association link throught intermediate class *class2*. Full path:

  clazz1.clazz2.clazz3

- Path to children of some class without direct link from other some class to this class.

- Recursive paths:

  clazz*

  While object not equals nil (if list, while list is not empty).

Our version of path-expression

- Name of attribute into the end of path:

  clazz.attr

  Results are values of this attribute.
- Name of attribute as intermediate piece of path.

  clazz.attr.clazz

  Type of this attribute must be class from domain.
- Object as piece of path:

  clazz1 (clazz2.$.clazz3)

How to laconic query text?

- exclude links between objects;
- using laconic names for classes of objects;
- default property for classes;
- complex conditions;
- user function.

Implementation

Query language

- selecting objects by class name;
- selecting values of objects;
- filtering due to specified conditions;
- selecting objects which are linked;
- union results of selecting;
- sorting (by objects, value of properties or user functions);
- calling user function;
- recursive queries;
- subqueries.

Query language: example

- Selecting objects:

~~select b from Building as b~~
building

- Selecting attributes:

~~select b.name, b.description from Building as b~~
building[name description]

- Filtering:

~~select b from Building as b where b.name='s-name'~~
building#(name="s-name")

- Sorting:

~~select b from Building as b order by b.name desc~~
{↑name}building

Eleminating links

- Selecting objects which are linked:

~~select b, f.rooms from Building as b join left b.floors as f~~
building (room)

- Nesting is not limit:

building (room (floor (device (simpleou))))

- Selecting multible objects of different classes:

building, room
building (room, device)
building (room, device (simpleou))

Query language: example

- Using laconic class name:

~~select li from LinkInterface as li~~
li

- Default property:

room#(floor=1)

- Simplification complex conditions:

~~floor#(number=1 || number=2)~~
floor#(number=(1 || 2))

Handling selecting objects

- Calling user function:

```
device@traffic
```

- Parameter — list of elements:

```
device@@traffic
```

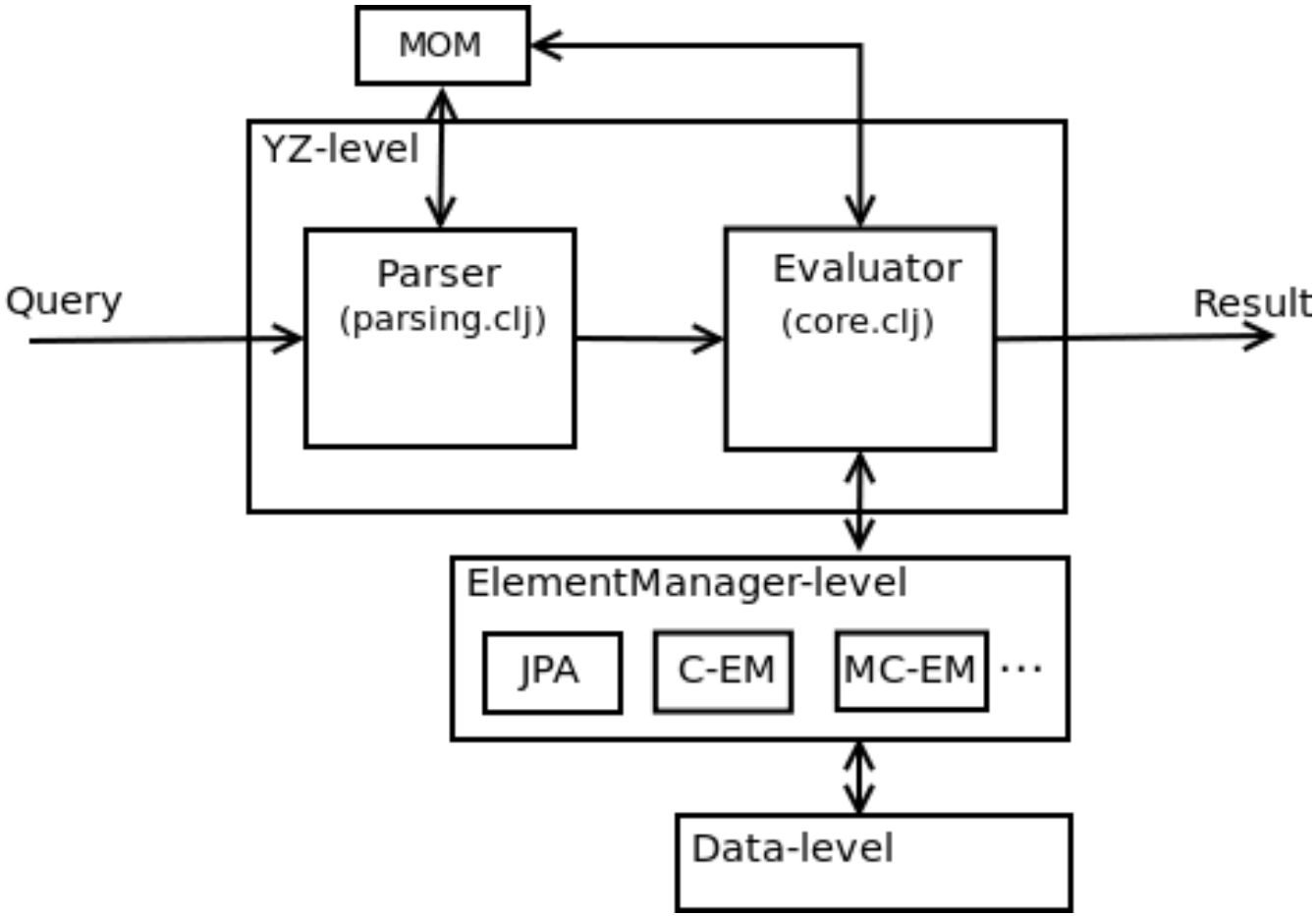
- Parameters of function (first — is result of query):

```
device@(traffic "01.10.2011" "10.10.2011")
```

- Calling function for each element from result of query:

```
device@@(traffic %)
```

Architecture



Example of queries into Nest

- builging (floor)

Building	Floor
Building ТК	Floor 1
	Floor 2
Building ГК	Floor 1
	Floor 2
	Floor 3
Building Общежитие№1	

- building (floor (room))

Building	Floor	Room
Building ТК	Floor 1	Room 100
		Room ТК
	Floor 2	
Building ГК	Floor 1	Room 149
		Room 146
		Room 110
	Floor 2	Room 215
		Room 217
	Floor 3	Room 333
Building Общежитие№1		

- building (floor (room (device)))

Building	Floor	Room	Device
Building ТК	Floor 1	Room 100	
		Room ТК	
	Floor 2		
Building ГК	Floor 1	Room 149	
		Room 146	Device
			Device
		Room 110	
	Floor 2	Room 215	
		Room 217	
	Floor 3	Room 333	
Building Об...			

Example of queries into Nest

- builging, floor

Building, Floor
Building ТК
Building ГК
Building Общежитие№1
Floor 1
Floor 2
Floor 1
Floor 2
Floor 3

- building (floor, room)

Building	Floor, Room
Building ТК	Floor 1
	Floor 2
	Room 100
	Room ТК
Building ГК	Floor 1
	Floor 2
	Floor 3
	Room 149
	Room 146
	Room 110
	Room 215
	Room 217
	Room 333
Buildign Общежитие№1	

- building (floor, room, device)

Building	Floor, Room, Device
Building ТК	Floor 1
	Floor 2
	Room 100
	Room ТК
Builgind ГК	Floor 1
	Floor 2
	Floor 3
	Room 149
	Room 146
	Room 110
	Room 215
	Room 217
	Room 333
	Device
	Device
Building Общежити№1	

GUI of Query Nestling