Artificial Intelligence II

2013/2014 - Prof: Daniele Nardi, Joachim Hertzberg

Exercitation 1 - Roberto Capobianco

Prolog, Inference Control, Meta-Programming, Semantic Maps

Contact Info

- Name: Roberto Capobianco
- Room: B121 (first floor)
- E-mail: capobianco@dis.uniroma1.it
- Web-page: https://sites.google.com/a/dis.uniroma1.it/capobianco/

Software

From the Artificial Intelligence I course:

- SWI-Prolog: http://www.swi-prolog.org/

Useful, but not needed:

- PrettyProlog: http://alessiostalla.altervista.org/software/prettyprolog/PrettyProlog.jar

  an interpreter for a subset of Prolog, written in Java, oriented towards didactical use: it features a GUI which allows to examine the inner functioning of the interpreter (stack, construction of the SLD tree)

Inference Control and Meta-Programming (Part I)

Inference Control in Prolog (Recap)

- **Program Specification;**
- **Clause Ordering;**
- **Conjunct Ordering;**
- Cut to Control Backtracking;

The structure of the program depends on what you want to obtain, since the order of the clauses makes the difference (top-down), as well as on the order of the conjuncts (left-right);

A Simple Knowledge Base

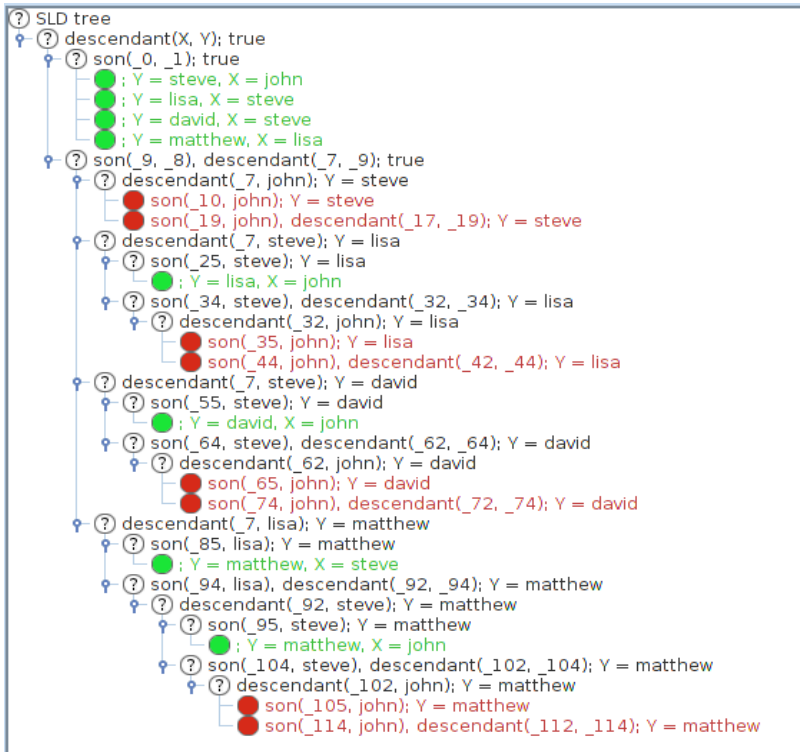
```
son(john, steve).
son(steve, lisa).
son(steve, david).
son(lisa, matthew).
```

i.e., son(X,Y): Y is a son of X.

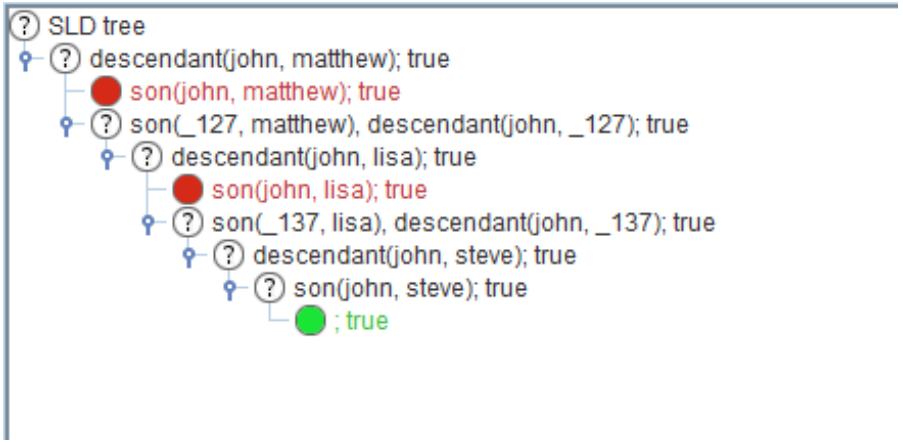
Program Specification (1)

```
descendant(X,Y):-son(X,Y).
descendant(X,Y):-son(Z,Y),descendant(X,Z).
```

descendant(X,Y).



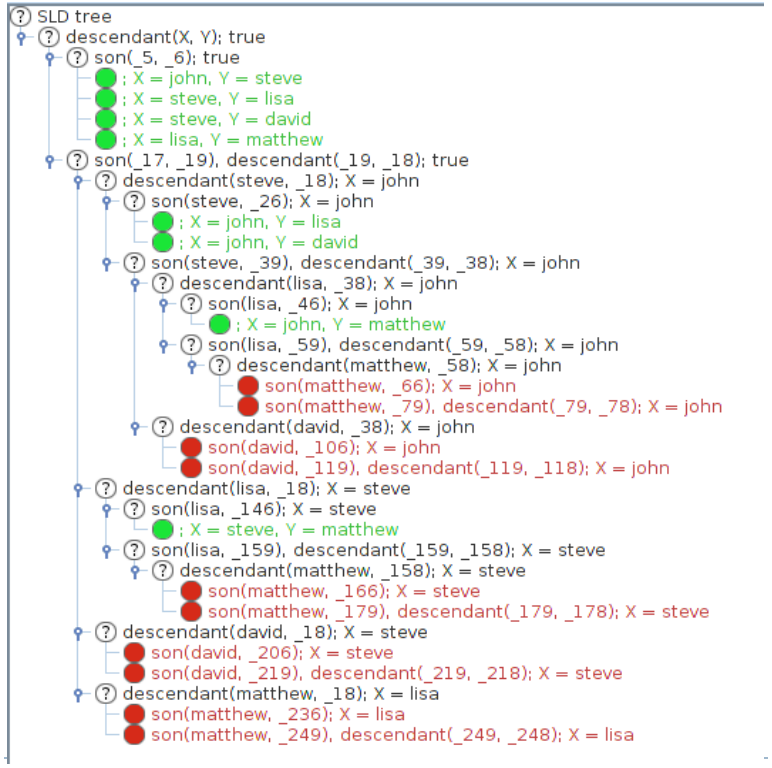
descendant(john, matthew).



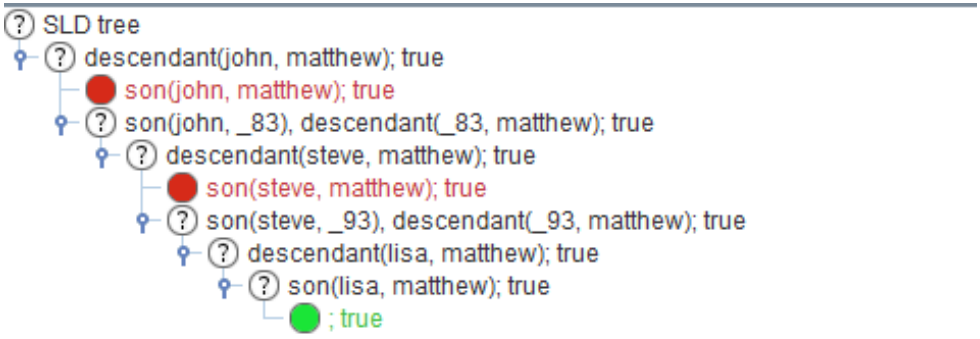
Program Specification (2)

```
descendant(X,Y):-son(X,Y).
descendant(X,Y):-son(X,Z),descendant(Z,Y).
```

descendant(X,Y).



descendant(john, matthew).



Program Specification (3)

descendant(X,Y):-son(X,Y).

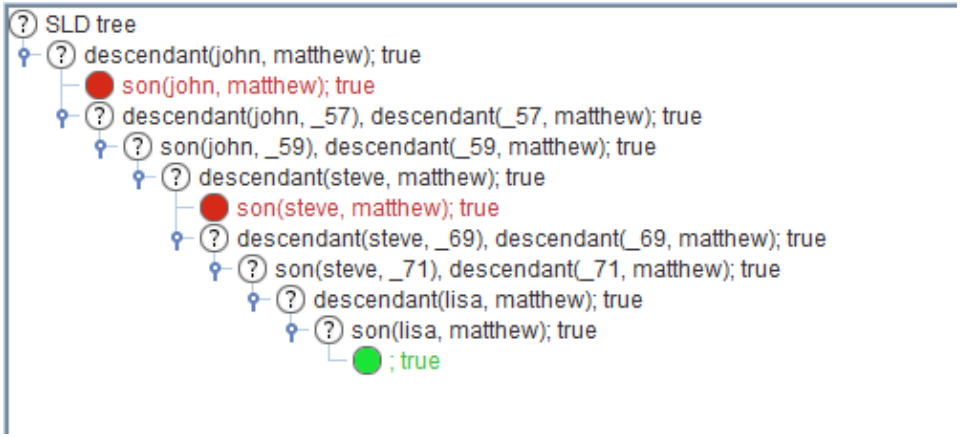
descendant(X,Y):-descendant(X,Z),descendant(Z,Y).

descendant(X,Y).

Let's have a look at the trace…

"Out of local stack"

descendant(john, matthew).



Conjunct & Clause Ordering

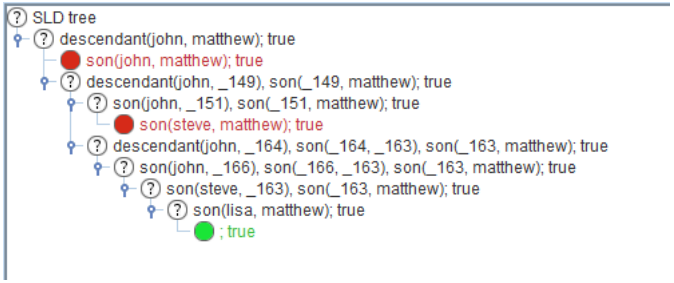
- **Conjunct ordering:**

```
descendant(X,Y):- son(X,Y).
descendant(X,Y):- descendant(X,Z),son(Z,Y).
```

descendant(X,Y).

"Out of local stack"

descendant(john, matthew).



- **Clause ordering:**

```
descendant(X,Y):- descendant(X,Z),son(Z,Y).
descendant(X,Y):- son(X,Y).
```

"Out of local stack"

Cut in Prolog (Recap)

- Program Specification;
- Clause Ordering;
- Conjunct Ordering;
- **Cut to Control Backtracking;**

The cut always succeeds, but it cannot be backtracked.

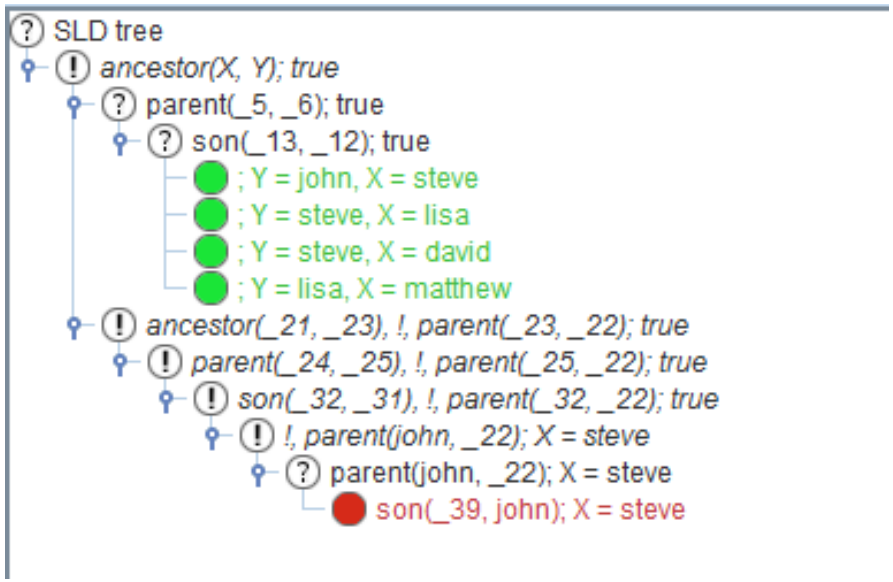
It is often used to avoid additional solutions and/or additional computations.

A green cut doesn't affect the solution of the program

A red cut… is not a green cut

Cut example

```
parent(X,Y):- son(Y,X).
ancestor(X,Y):- parent(X,Y).
ancestor(X,Y):- ancestor(X,Z),!,parent(Z,Y).
```



Prolog Meta-Interpreters (Recap)

- *Meta-Interpreters*: interpreters written for the same language in which they are written (in this case, Prolog);
- The program which is being interpreted by the meta-interpreter is called the *object program;*
- Prolog itself provides a way to access the clause database using the clause/2 predicate.
- clause(H,B) finds a clause in Prolog program with head that unifies with H and body B (if there is no body, then Body=true).

The Simplest Meta-Interpreter

The simplest Prolog meta-interpreter is the following program:

```
solve(Goal):-call(Goal).
```

However, there is not advantage of using such meta-interpreter as it immediately calls Prolog interpreter.

Vanilla Interpreter

- Much more popular is "vanilla" meta-interpreter that uses Prolog's built-in unification but enables access to search engine which can be easily modified (e.g., it is possible to change the order of goals' execution)

```
solve1(true).
solve1((A,B)):- solve1(A), solve1(B).
solve1(A):- clause(A,B), solve1(B).
```

Let's trace…

Other Interpreters (1)

```
solvept(true,true):- !.
solvept((A,B),(ProofA,ProofB)):-
     !, solvept(A,ProofA), solvept(B,ProofB).
solvept(A,(A:-Proof)):-
     clause(A,B), solvept(B,Proof).
```

Interpreter which builds the proof tree.

Other Interpreters (2)

```
%In this mini-interpreter, goals and clauses are represented as ordinary Prolog data
%structures (i.e. terms).
%Terms  representing  clauses are specified using the unary predicate my_clause,
%e.g.

my_clause( (grandparent(X,Z):-parent(X,Y),parent(Y,Z)) ).

%A unit clause will be represented by a term such as

my_clause( (parent(john,mary) :- true) ).

%The mini-interpreter consists of four clauses:

execute(true) :-!.
execute((P,Q)) :- !, execute(P), execute(Q).
execute(P) :- my_clause((P:-Q)), execute(Q).
execute(P) :- P.

%The last clause enables the mini-interpreter to cope  with  calls  to  ordinary
%Prolog predicates, e.g.  evaluable predicates.
```

Other Interpreters (3)

```
clever(joe).
handsome(joe).

rule([handsome(joe),clever(joe)],(hasgirlfri
end(joe))).

run:-
call(rule(X,Y)),findall(A,(member(A,X),call(
A)),L),
     length(X,Rulelength),
     length(L,Rulelength),
     assert(Y),print(Y).
```

Semantic Maps for Robots (Part II)

Semantic Mapping

- The term semantic mapping is used to denote the process that allows the robot to enrich the map used for navigation with an explicit representation of the knowledge about the environment
- Semantic maps allow the robot to…

✓ …perform reasoning over environments, objects and properties

✓ …communicate with humans, understanding complex commands

✓ …act in a complex way

Semantic Mapping Techniques

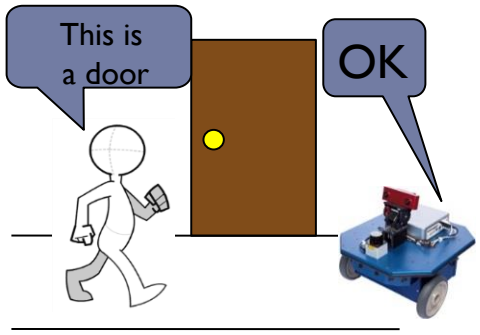
- fully automated mapping
  - extraction of attributes of rooms
  - doorways detection
  - classification and clustering
  - object recognition

- human augmented mapping
  - human-in-the-loop
  - multi-modal interaction

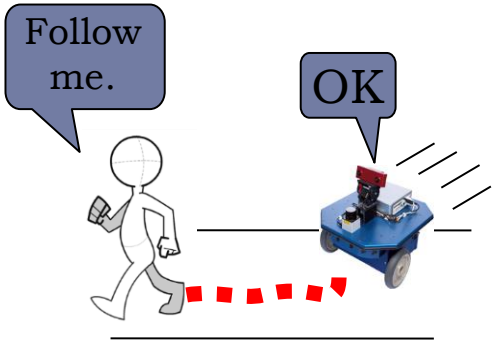
- lack of details in the representation
- little involvement of the user
- ambiguities and inconsistencies

Human-in-the-loop

Labeling

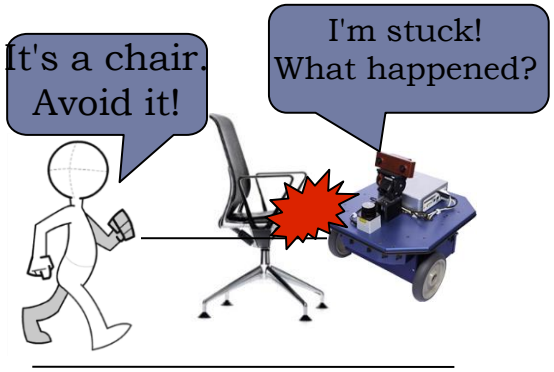


Instructing

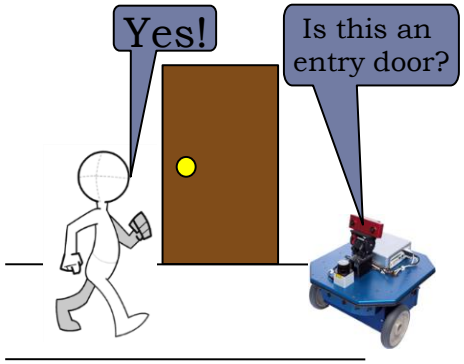


PROBLEM:
establish a relationship between symbols and elements of the operational environment (**symbol grounding**).

Re-planning



Disambiguating



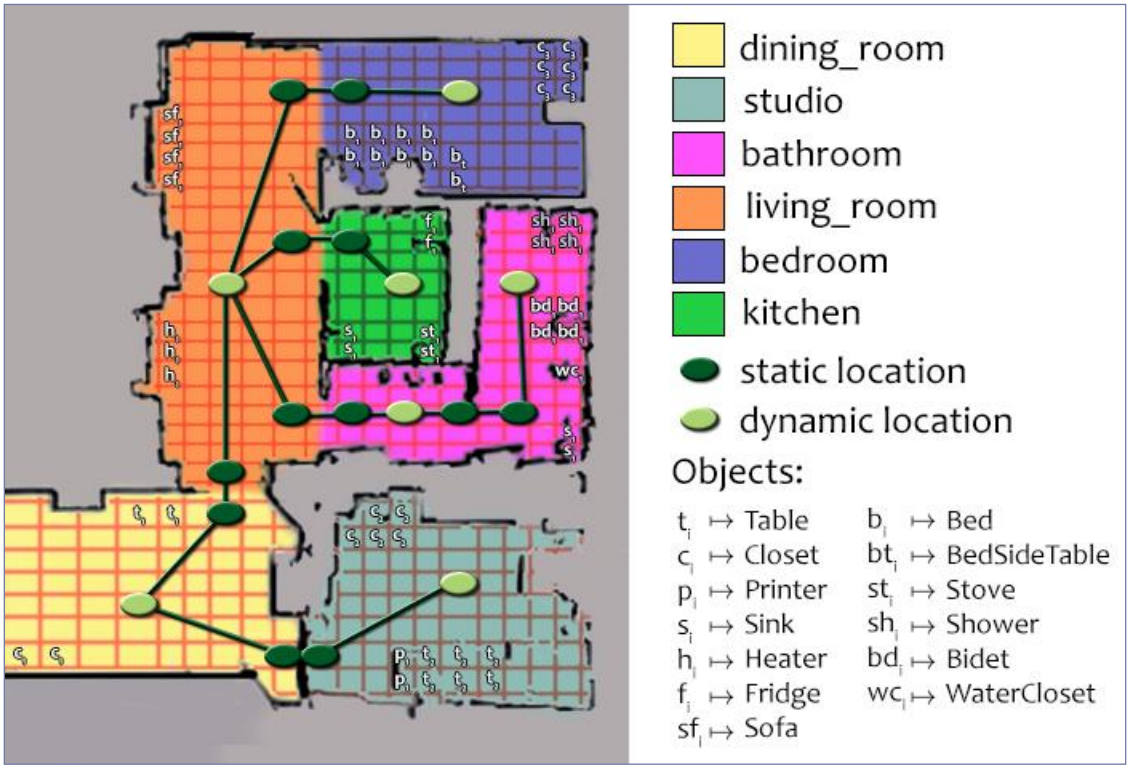
SOLUTION:
Symbiotic autonomy paradigm

Knowledge Representation

- Domain knowledge:
  - hierarchy of concepts;
  - general knowledge about environments;

- World knowledge:
  - specific knowledge about the environment;
  - instance signatures;
  - metric map:
  - grid map;
  - cell map;
  - topological graph;

World Knowledge



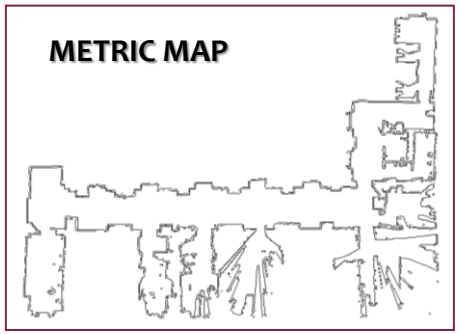
Instance Signatures Database

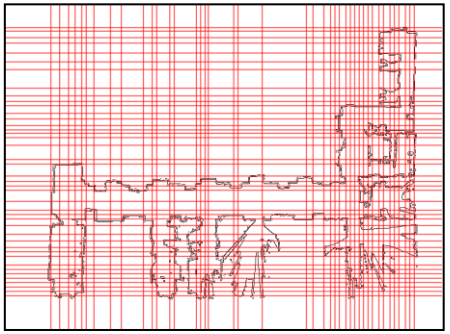
- It is a database of structured data, where each instance has a unique label *I*, an associated concept *C* such that $I \rightarrow C$, and a set of attribute-value properties.

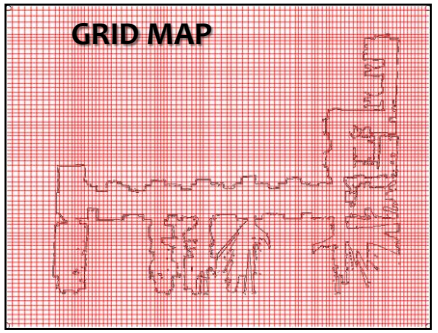
<fridge1 → Fridge, (position = < x,y,θ >, color = white, open = false, model=13061015340)>

Grid Map

- Qualitative representation of the environment;
- Based on the lines extracted from the metric map;
- Consistent both at the metric and symbolic layers;

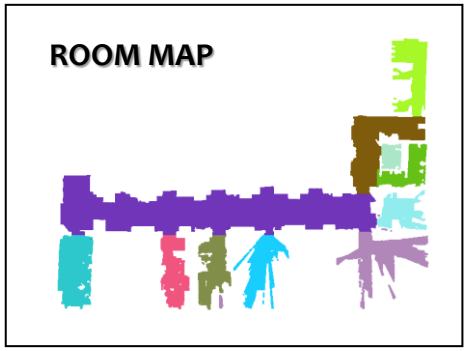


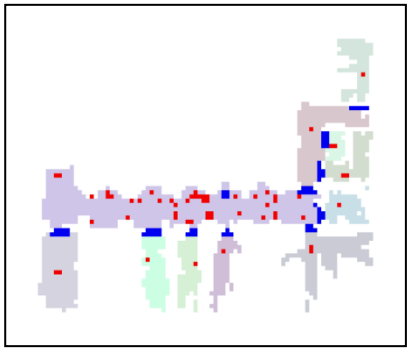




Cell Map & Semantic Map

- Generation of a Room Map on the metric map from a set of tags about areas and doors;
- Generation of a Semantic Map from the Room Map and a set of tags about objects;





Prolog Representation (Semantic Map)

```
object(Id, XCoord, YCoord, Properties).
objectType(Id, Type).
```

For example, the knowledge of a white plug located in the cell with grid map coordinates 45, 67 belonging to the corridor area will be represented with the three predicates

```
celIsPartOf(45, 67, corridor).
objectType(plug1, plug).
object(plug1, 45, 67, color-white).
```

Search a specific object *ObjectAtom* of type *ObjectType* in room *Room:*

```
objectOf(ObjectType,Room,ObjectAtom) :-
objectType(ObjectAtom,ObjectType), object(ObjectAtom, X, Y,
_), cellIsPartOf(X,Y,Room).
```

Prolog Representation (Graph)

```
dynamicNode(Id, XCoord, YCoord).
staticNode(Id, XCoord, YCoord).
arc(Id1, Id2).
```

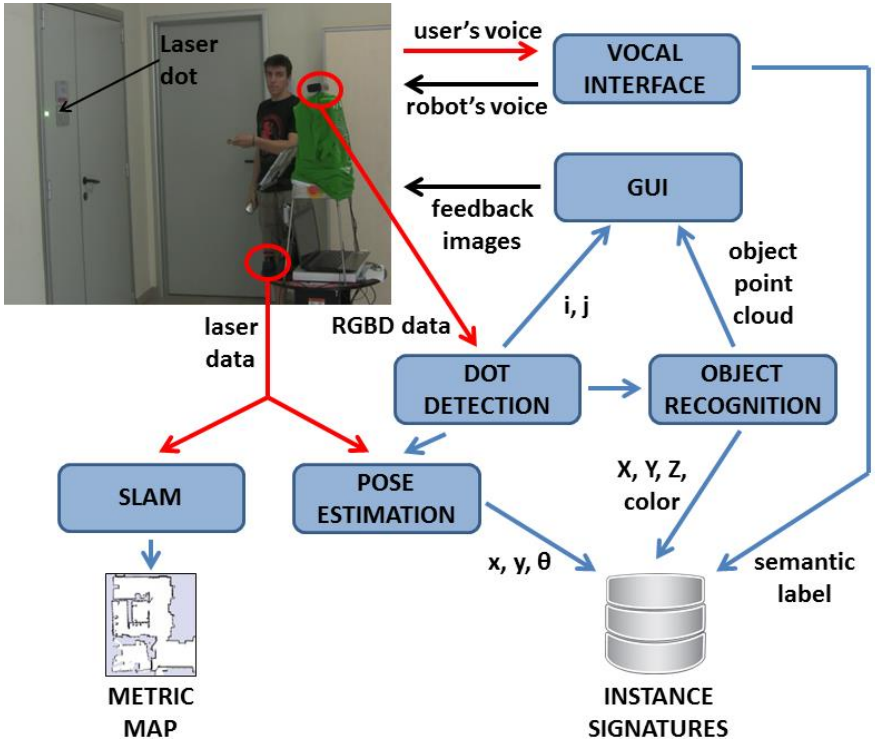
Search on the graph:

```
connected(X,Y) :- edge(X,Y) ; edge(Y,X); X==Y.

path(A,B,Path) :- travel(A,B,[A],Q), reverse(Q,Path).

travel(A,B,P,[B|P]) :- connected(A,B).
travel(A,B,Visited,Path) :- connected(A,C), C \== B,
\+member(C,Visited), travel(C,B,[C|Visited],Path).
```

Knowledge Acquisition



Use of the Semantic Map (1)

User: *"Go to the socket.".*

Robot: *"There are many sockets. Which one do you mean?"*

User: *"The black one close to the emergency door."*

Robot: *"OK. I am going to the black socket."*

User: *"Go to the book closet."*

Robot : *"I don't know where the book closet is."*

"But there are some closets. Which one do you mean?"

User : *"Go to the closet near the PhD room."*

Robot : *"OK. I am going to the closet near the PhD room."*

Use of the Semantic Map (2)

- disambiguation handling:
  - synonyms
  - general/specific concepts
  - spatial relations
  - properties
- consistency management:
  - user-based consistency managements
  - deals with objects in the same position

Thank you!