european social fund in the czech republic
EUROPEAN UNION
MINISTRY OF EDUCATION, YOUTH AND SPORTS
OP Education for Competitiveness
INVESTMENTS IN EDUCATION DEVELOPMENT

1 Inductive inference in predicate logic

We still use Prolog notation for both the data and the learned hypotheses.

Exercise 1.1: Let us have the three-digit numbers categorization (the same as in the lecture):

example	139	319	854	468	349	561	756	789	987	256	189	354
classif.	+	−	−	+	+	−	−	+	−	+	+	−

a) specify the domain knowledge

b) draw the relevant part of a specialization graph

c) compare the learned hypothesis with the decision tree for the original task

d) for the correct and complete solution: what (minimal) number of examples is needed for learning the decision tree? And how many for learning the Prolog hypothesis?

Solution 1.1:

a) We know that the classification corresponds to the ordering of digits in numbers, we assume that zero is not used (it is not present in the examples). We will represent the numbers as three separated digits.
It is possible to use the built-in operator `</2` as a domain knowledge, however, we will define our own predicate `lt` instead. It is defined as follows:
```
lt(1,2).  lt(2,3).  ... lt(8,9).
lt(1,3).  lt(2,4).  ...
...
```

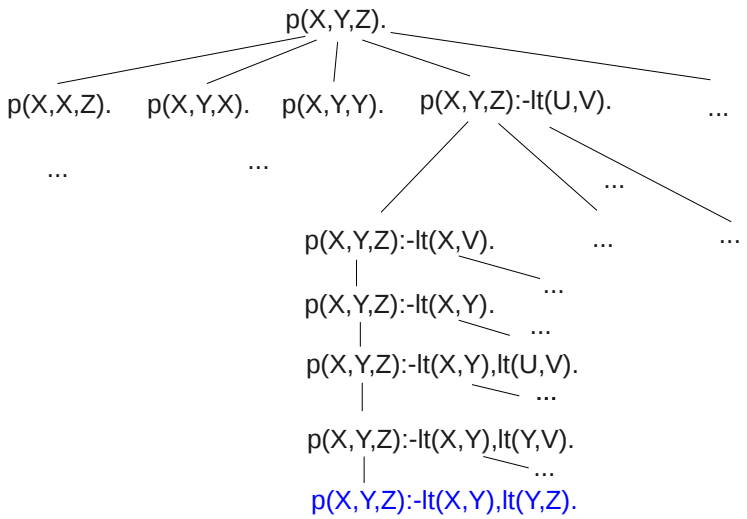
b) We will use four specialization operators. If we use them to specialize the clause `p(X,Y)`, we get the following results:

- unification of two variables
  `p(X,X).`
- adding a subgoal from the domain knowledge (or the predicate currently being derived)
  `p(X,Y) :- p(U,V).`
  or
  `p(X,Y) :- r(Z). % domain knowledge: r/1`
- consistent substitution of a variable by a constant
  `p(X,[]).`
  Usually when we know which variable is useful to substitute with which constants (it can be observed in the learning data).

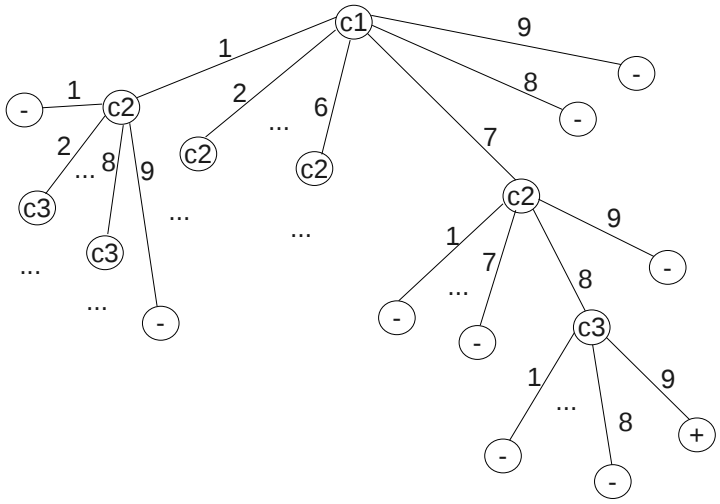
- consistent substitution of a variable by a most general term
  `p(X,[H|T]).`
  Conditions similar to the previous specialization operator apply.

The graph of specializations has a similar structure as in propositional logic. However, it is typically much larger. We usually draw only its relevant part: a subtree that shows the steps which lead to the final hypothesis.

The graph is constructed according to the learning data. Only paths that cover at least one positive example are taken into account. When a node covers only positive examples, it is added into the hypothesis and it is not further specialized. The process ends when the hypothesis covers all positive examples. Algorithms use various optimisation techniques for recognition of potentially promising paths.



c) Decision trees are also suitable for solving classification tasks. We try to demonstrate that for this kind of tasks inductive inference in predicate logic is much more effective.



d) We need all the positive examples (each of them has an exclusive path in the tree) for learning the decision tree. There are $\binom{9}{3} = 84$ of them.
Inductive inference needs one positive example, e.g.
`p(1,2,3).`
and negative examples that prevent searching in all improper paths, e.g.
`p(1,1,1). p(1,3,2). p(2,1,3). p(2,3,1). p(3,2,1). p(3,1,2).`

Exercise 1.2: Represent the following task using predicate logic (it has already been represented in propositional logic).
There are the following data with three attributes classified into two classes `true/false`:

`Size` $\in$ {`small, medium, large`},
`Color` $\in$ {`red, blue, green`},
`Shape` $\in$ {`square, circle, triangle`}

small	red	triangle	true
small	green	triangle	true
large	red	triangle	false
small	blue	circle	false

Find one or more proper specializations and one or more proper generalizatons of the following clauses. Do not specialize the variable `Id`.

a) `p(Id) :- size(Id,large), color(Id,red).`

b) `p(Id) :- color(Id,red).`

c) `p(Id) :- size(Id,large), color(Id,red), shape(Id,circle).`

d) `p(Id).`

e) Find all specializations of the clause `p(Id) :- color(Id,red).` that cover the example `<large,red,square>`.

f) Find out whether (and how) the formulas from items a)–d) are in the generalization/specialization relation.

Solution 1.2: Domain knowledge: predicates `size/2, color/2, shape/2`.
Training data:

```
size(1,small).  color(1,red).  shape(1,triangle).
size(2,small).  color(2,green).  shape(2,triangle).
...
```

Positive examples: `p(1). p(2).`
Negative examples: `p(3). p(4).`

a) there are considerably more specializations than in propositional logic, some of them (from one specialization path) follows:

```
p(Id) :- size(Id,large), color(Id,red), shape(X,Y).
p(Id) :- size(Id,large), color(Id,red), shape(Id,Y).
p(Id) :- size(Id,large), color(Id,red), shape(Id,circle).
...
p(Id) :- false.
```

b) e.g. `p(Id) :- color(Id,red), size(Id,small).`

c) e.g. `p(Id) :- false.`

d) e.g. `p(Id) :- color(Id,green).`

e) e.g.

```
p(Id) :- color(Id,red), size(Id,large).
p(Id) :- color(Id,red), shape(Id,square).
p(Id) :- color(Id,red), size(Id,large), shape(Id,square).
```

f) No pair is in the minimal generalization or minimal specialization relation. There are many other generalization/specialization relations, e.g. d) is a generalization of all other clauses.

Exercise 1.3:

a) Draw the specialization graph for the predicate `member/2`.

b) Describe the changes in the graph for the predicate `last/2`.

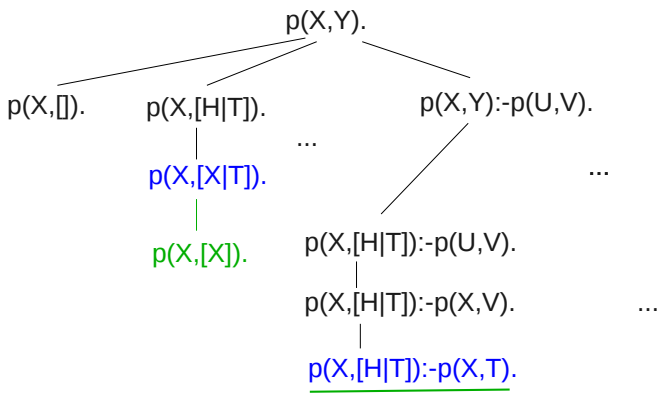
c) Write lists of training examples for these predicates.

Solution 1.3:

```
member(X,[X|_]).
member(X,[_|T]):- member(X,T).

last(X,[X]).
last(X,[_|T]):- last(X,T).
```

The final theory for `member` is marked blue. The changes and the final theory for `last` are marked green.
Positive examples for `member`: `p(a,[a]).` `p(a,[b,a]).` `p(a,[a,b]).`
Negative examples for `member`: `p(a,[]).` `p(a,[b,c]).`
Training examples for `last` are the same, only `p(a,[a,b]).` moves from positive examples to negative ones.



Exercise 1.4: Describe the construction of a specialization graph for `reverse/2`. The domain knowledge consists of the predicate `append/3`.

Solution 1.4:

```
append([],L,L).
append([H|T],L,[H|T2]):-append(T,L,T2).

reverse([],[]).
reverse([H|T],R):-reverse(T,Y),append(Y,[H],R).
```

The construction of the graph is similar to the previous task.