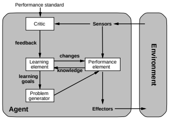
Clinical Decision Support Systems, 23/24

Inês Dutra and Pedro Rodrigues
DCC-FCUP & MEDCIDS-FMUP
ines@dcc.fc.up.pt, pprodrigues@med.up.pt

March 1st, 2024



(Multi) Relational Data Mining

- Inductive Logic Programming (ILP)
- Probabilistic Reasoning – PR
- ILP + PR = SRL
- Statistical Relational Learning

Inductive Logic Programming

Deductive Reasoning

$T \cup B \models E$

T		B		E
parent(X,Y) :- mother(X,Y) parent(X,Y) :- father(X,Y)	$\cup$	mother(penelope,victoria) mother(penelope,artur) father(christopher,victoria) father(christopher,artur)	$\models$	parent(penelope,victoria) parent(penelope,artur) parent(christopher,victoria) parent(christopher,artur)

Inductive Reasoning

$E \cup B \models T$

E		B		T
parent(penelope,victoria) parent(penelope,artur) parent(christopher,victoria) parent(christopher,artur)	$\cup$	mother(penelope,victoria) mother(penelope,artur) father(christopher,victoria) father(christopher,artur)	$\models$	parent(X,Y) :- mother(X,Y) parent(X,Y) :- father(X,Y)

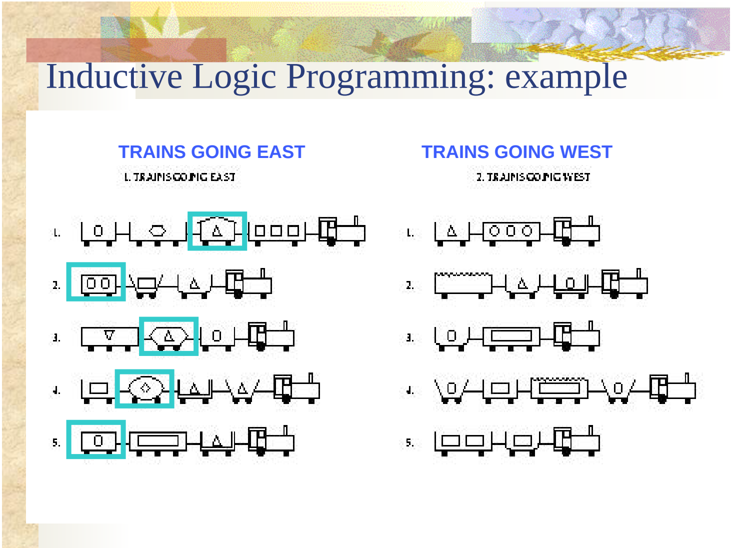
Inductive Logic Programming

Inductive Logic Programming

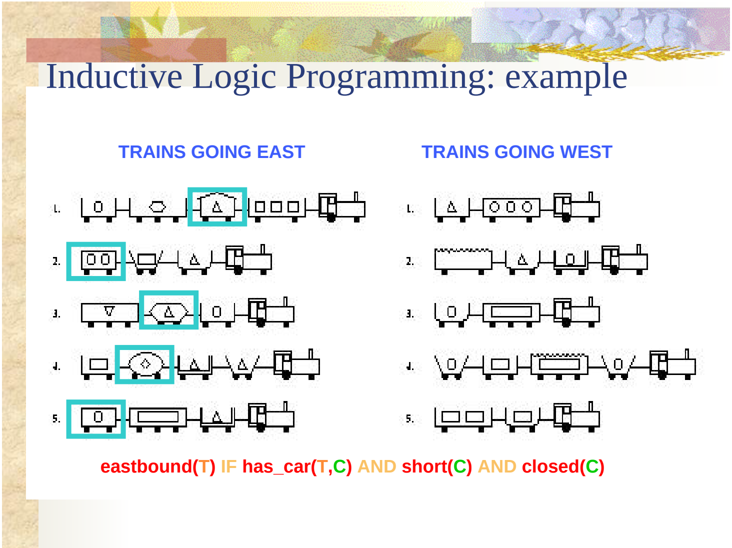
Inductive Logic Programming: example

short(car_12).
closed(car_12).
long(car_11).
long(car_13).
short(car_14).
open_car(car_11).
open_car(car_13).
open_car(car_14).
shape(car_11,rectangle).
shape(car_12,rectangle).
shape(car_13,rectangle).
shape(car_14,rectangle).

load(car_11,rectangle,3).
load(car_12,triangle,1).
load(car_13,hexagon,1).
load(car_14,circle,1).
wheels(car_11,2).
wheels(car_12,2).
wheels(car_13,3).
wheels(car_14,2).
has_car(east1,car_11).
has_car(east1,car_12).
has_car(east1,car_13).
has_car(east1,car_14).


Inductive Logic Programming: example
TRAINS GOING EAST
TRAINS GOING WEST
1. TRAINS GOING EAST
2. TRAINS GOING WEST
1.
2.
3.
4.
5.
1.
2.
3.
4.
5.

Inductive Logic Programming



Inductive Logic Programming

Another example: extracting knowledge from mammogram annotations

```
is_malignant(A)  if
    'BIRADS_category'(A,b5),'MassPAO'(A,present),
    'Age'(A,age6570),
    previous_finding(A,B), 'MassesShape'(B,none),
   'Calc_Punctate'(B,notPresent),
    previous_finding(A,C), 'BIRADS_category'(C,b3).
```

The learned rule above says:

A is classified as BI-RADS 5 AND had a mass present
in a patient who:
was between the ages of 65 and 70
had two prior mammograms (B, C)
AND prior mammogram (B):
had no mass shape described
had no punctate calcifications
AND prior mammogram (C) was classified as BI-RADS 3

BI-RADS: **B**reast **I**maging **R**eporting **A**nd **D**ata **S**ystem

Inductive Logic Programming

- More formally:
- Given:
  - A set of examples **e** (observations, cases, instances) labelled as positive or negative (class **c**)
  - A language
  - Possibly, a set of constraints
- Find:
  - A hypothesis h, such that $\mathbf{h}(\mathbf{e}_i) = \mathbf{c}_i$
  - For most examples

Inductive Logic Programming

Inductive Logic Programming

- Advantages:
  - Utilization of a language that is easy to interpret
  - More concise classifiers
  - More powerful representation: relations
- Disadvantages:
  - Very large search space
  - Non-probabilistic classification

Properties

- Prior satisfiability

$$B \wedge E^- \not\models \Box$$

(H is not a consequence of B and E-)

- Posterior sufficiency

$$B \wedge H \models E^+$$

(H allows to explain E+ relative to B)

- Posterior satisfiability

$$B \wedge H \wedge E^- \not\models \Box$$

(B and H are consistente with E-)

- Prior necessity

$$B \not\models E+$$

(some e+ must be false relative to the model found for B)

ILP: A Common Approach

- Use a greedy covering algorithm.
  - Repeat while some positive examples remain uncovered (not entailed):
    - Find a *good clause* (one that covers as many positive examples as possible but no/few negatives).
    - Add that clause to the current theory, and remove the positive examples that it covers.
- ILP algorithms use this approach but vary in their method for finding a *good clause*.

Some ILP Systems

- PROGOL, ALEPH (top-down): saturates first uncovered positive example, and then performs top-down admissible search of the lattice above this saturated example.
- GOLEM (bottom-up), FOIL (top-down), LINUS/DINUS.
- Tilde, Claudien, IndLog, ...

ILP Saturation

- Consists of building a *bottom clause* (seed)
- Incorporates background knowledge to an atomic formula
- Example: (gene that codes for a protein responsible for metabolism)

```
metabolism(A) :-
    essential(A,'Non-Essential'), motif(A,'PS00510'), chromosome(A,'14'),
    interaction(A,B,C,E),
    essential(B,'Non-Essential'), motif(B,'PS00188'), chromosome(B,'2'),
    interaction(A,F,D,G),
    intertype(C,'Genetic'), intertype(D,?),
    interaction(B,A,C,E),
    interaction(B,H,C,I),
    interaction(F,A,D,G),
    interaction(H,B,C,I), interaction(H,_,_,_).
```

Inductive Logic Programming

ILP: Aleph

- Procedure to extract theories from examples
- Complete (branch-and-bound) search for best clause in the ***whole*** space
- Search subject to several user control settings
  - Max clause length
  - Max chaining length
  - Minacc
  - Max nodes
  - Search strategy, etc.

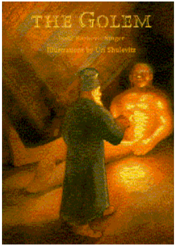
ILP: Aleph

- Aleph
  - Desenvolvido na Universidade de Oxford por Ashwin Srinivasan

http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/

ILP: Aleph

Then the Rabbi said,
"Golem, you have not been completely formed, but I am about to finish you now…You will do as I will tell you."
Saying these words, Rabbi Leib finished engraving the letter Aleph. Immediately the golem began to rise.



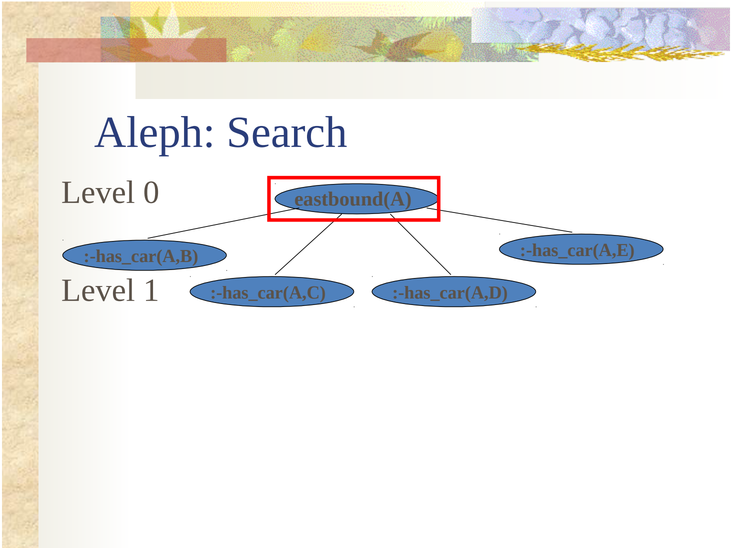
Aleph: algorithm

- Initial State:
  - Examples or observations
  - Descriptions: background knowledge (BK)
- Final State: hypothesis or theory or model
- Transitions: intermediate hypotheses

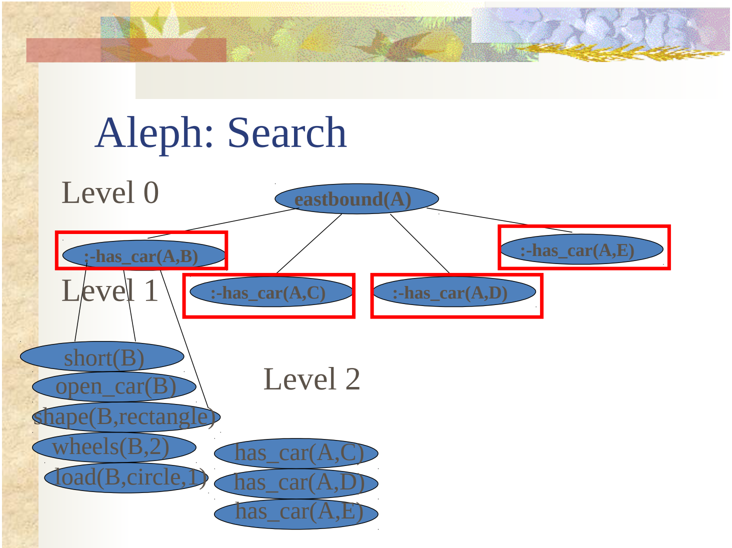
Aleph: algorithm

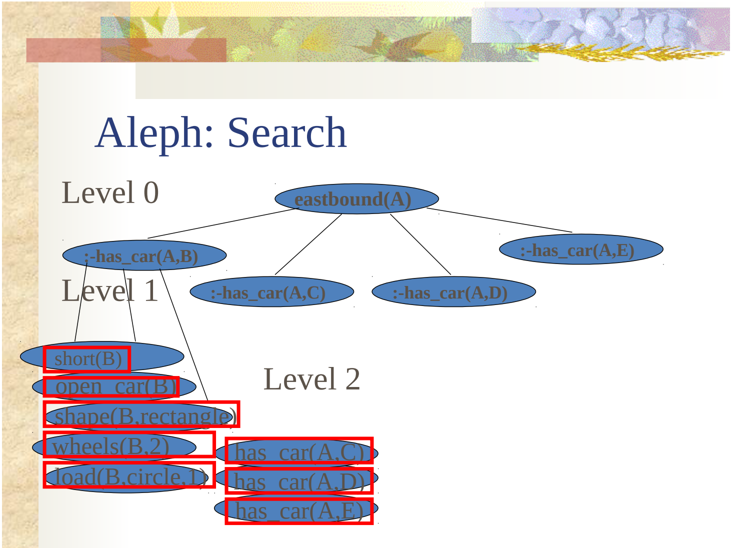
- Select example
- Build most-specific-clause (**bottom clause**)
- Search. Find a clause more general than the bottom clause
- Remove redundant. The clause with the best score is added to the current theory, and all examples made redundant are removed. This step is sometimes called the "**cover removal**" step. Note here that the best clause may make clauses other than the examples redundant
- Return to first step

Inductive Logic Programming



Inductive Logic Programming

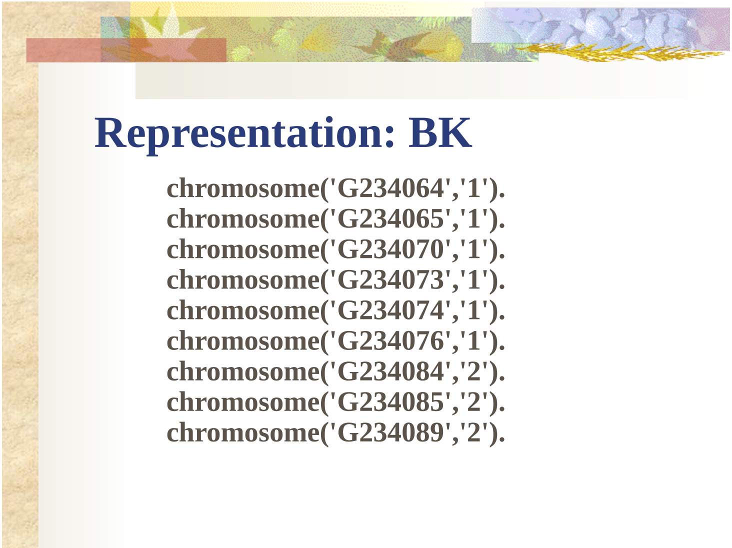



Aleph: Search
Level 0
eastbound(A)
:-has_car(A,B)
:-has_car(A,C)
:-has_car(A,D)
:-has_car(A,E)
Level 1
short(B)
open car(B)
shape(B,rectangle)
wheels(B,2)
load(B,circle,1)
Level 2
has car(A,C)
has car(A,D)
has_car(A,E)

Aleph: Knowledge Representation

Input Files: Prolog Syntax

dtp.b: BK
dtp.f: pos examples
dtp.n: neg examples


Representation: BK
chromosome('G234064','1').
chromosome('G234065','1').
chromosome('G234070','1').
chromosome('G234073','1').
chromosome('G234074','1').
chromosome('G234076','1').
chromosome('G234084','2').
chromosome('G234085','2').
chromosome('G234089','2').

Representation: BK

```
interaction('G234062','G235011','Physical',?).
interaction('G234064','G234126','Genetic-
               Physical','0.9141').
interaction('G234064','G235065','Genetic-
               Physical','0.7515').
interaction('G234064','G235571','Physical','0.9691').
interaction('G234065','G234073','Physical','0.7492').
interaction('G234065','G235042','Physical','-0.4659').
```

Representation: Examples

```
metabolism('G239098').
metabolism('G234980').
metabolism('G235245').
metabolism('G234108').
metabolism('G238387').
metabolism('G240504').
metabolism('G236733').
```

Example of clause learned

```
metabolism(A) :-
        chromosome(A,'15'),
        interaction(A,B,_,_),
        complex(B,'Transcription
    complexes/Transcriptosome').
```

A **and** ***B*** **are variables that represent genes**

Aleph: algorithm

- Example: Michalski´s trains

Aleph: algorithm

- Saturation (saturated / bottom clause):

eastbound(A) :-
has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
short(B), short(D), closed(D), long(C),
long(E), open_car(B), open_car(C), open_car(E),
shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
shape(E,rectangle),
wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
load(E,rectangle,3).

Aleph: algorithm

- Search: most general clause

eastbound(A) :-
has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
short(B), short(D), closed(D), long(C),
long(E), open_car(B), open_car(C), open_car(E),
shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
shape(E,rectangle),
wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
load(E,rectangle,3).

Aleph: algorithm

- Search: add possible descendants (candidate literals of level 1)

```
eastbound(A) :-
  has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
  short(B), short(D), closed(D), long(C),
  long(E), open_car(B), open_car(C), open_car(E),
  shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
  shape(E,rectangle),
  wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
  load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
  load(E,rectangle,3).
```

Aleph: algorithm

- Search: add possible descendants of level 2

```
eastbound(A) :-
  has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
  short(B), short(D), closed(D), long(C),
  long(E), open_car(B), open_car(C), open_car(E),
  shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
  shape(E,rectangle),
  wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
  load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
  load(E,rectangle,3).
```

Aleph: algorithm

- Search: second descendant of level 1

eastbound(A) :-
has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
short(B), short(D), closed(D), long(C),
long(E), open_car(B), open_car(C), open_car(E),
shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
shape(E,rectangle),
wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
load(E,rectangle,3).

Aleph: algorithm

- Search: descendants of second descendant…

eastbound(A) :-
has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
short(B), short(D), closed(D), long(C),
long(E), open_car(B), open_car(C), open_car(E),
shape(B,rectangle), shape(C,rectangle), shape(D,rectangle),
shape(E,rectangle),
wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
load(B,circle,1), load(C,hexagon,1), load(D,triangle,1),
load(E,rectangle,3).

Aleph: example of run

- aleph_trains

Aleph: how to run?

- You need to have a Prolog system
  - Yap: http://yap.sourceforge.net OU
  - SWI: http://www.**swi**-prolog.org
- Aleph: http://www.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/
- Files: .b, .f, .n
- To make things easier: everything in the same directory!

Aleph: Basic Commands

- read_all
- reduce
- induce

Aleph: Parameters

```
:- set(clauselength,5).
:- set(depth, 200).
:- set(i,3).
:- set(noise,0).
:- set(minacc,0.7).
:- set(nodes,1000000).
:- set(m,20).
:- set(evalfn,mestimate).
:- set(test_pos,'/u/dutra/Protein/prot_test_set.f').
:- set(test_neg,'/u/dutra/Protein/prot_test_set.n').
:- set(optimise_clauses,true).

:- set(record,true).
:- set(recordfile,'prot_train_set.out').
:- set(samplesize,0).
```

Strength estimate = (*support* + *m* * *prior*) / (*coverage* + *m*)

M → 0, strength → precision

Support = True positives

Coverage = True positives + false negatives

Inductive Logic Programming

Aleph: Modes and Types

```
:- modeh(1,eastbound(+train)).
:- modeb(1,short(+car)).
:- modeb(1,closed(+car)).
:- modeb(1,long(+car)).
:- modeb(1,open_car(+car)).
:- modeb(1,double(+car)).
:- modeb(1,jagged(+car)).
:- modeb(1,shape(+car,#shape)).
:- modeb(1,load(+car,#shape,#int)).
:- modeb(1,wheels(+car,#int)).
:- modeb(*,has_car(+train,-car)).
```

```
:- determination(eastbound/1,short/1).
:- determination(eastbound/1,closed/1).
:- determination(eastbound/1,long/1).
:- determination(eastbound/1,open_car/1).
:- determination(eastbound/1,double/1).
:- determination(eastbound/1,jagged/1).
:- determination(eastbound/1,shape/2).
:- determination(eastbound/1,wheels/2).
:- determination(eastbound/1,has_car/2).
:- determination(eastbound/1,load/3).
```

Aleph: Modes and Types

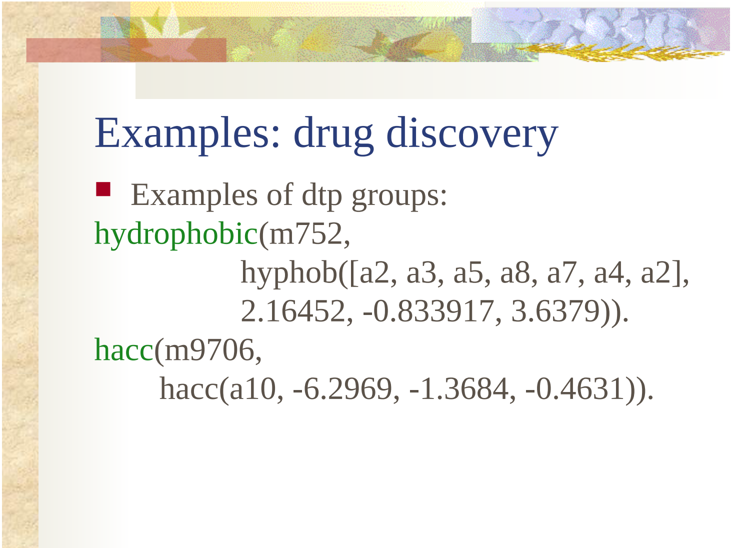
```
:- modeh(1,metabolism(+gene)).

:- modeb(1,essential(+gene,#essential)).
:- modeb(1,class(+gene,#class)).
:- modeb(1,complex(+gene,#complex)).
:- modeb(1,phenotype(+gene,#phenotype)).
:- modeb(1,motif(+gene,#motif)).
:- modeb(1,chromosome(+gene,#chromosome)).
:- modeb(*,gte(+number,#number)).
:- modeb(*,interaction(+gene,-gene,-intertype,-number)).
:- modeb(1,intertype(+intertype,#intertype)).
```

Inductive Logic Programming

Example: drug discovery using Aleph refinement operators

- Given:
  - Molecules active and inactive for dtp
  - Their description in terms of coordinates and bonds
- Find small structures that model active molecules


Examples: drug discovery
Examples of dtp groups:
hydrophobic(m752,
hyphob([a2, a3, a5, a8, a7, a4, a2],
2.16452, -0.833917, 3.6379)).
hacc(m9706,
hacc(a10, -6.2969, -1.3684, -0.4631)).

Example: drug discovery

- Utilisation of refinement operator

```
refine(false,Clause):-
   member(Point1, [hydrophobic(M,P1), hdonor(M,P1),halogen(M,P1),hacc(M,P1)]),
   member(Point2,[hydrophobic(M,P2),hdonor(M,P2),halogen(M,P2),hacc(M,P2)]),
   Clause = (active(M) :- Point1, Point2, dist(M,P1,P2,D1,E)).

refine(Clause1,Clause2):-
      Clause1 = (active(M) :- Point1,Point2, dist(M,P1,P2,D1,E)),     member(Point3,
      [hydrophobic(M,P3),hdonor(M,P3),halogen(M,P3),hacc(M,P3)]),
      Clause2 = (active(M) :- Point1, Point2, dist(M,P1,P2,D1,E),
                  Point3, dist(M,P1,P3,D2,E), dist(M,P2,P3,D3,E)).
```

- Reduce search space!!!