AUTONOMOUS DISCOVERY OF ABSTRACT CONCEPTS BY A ROBOT

Ivan Bratko
University of Ljubljana

This talk

- About robots that
  - Learn from observations
  - “Gain insights”

- Which methods of Machine Learning are suitable for gaining insights?

Scientific goal

Investigate mechanisms of
autonomous discovery
through experiments in
an agent’s environment

LEVELS OF ROBOT PROGRAMMING

- **Fixed sequence** of direct commands

  Limitation: Only works for exactly **predefined tasks**

- Define **model** of environment, robot **plans** actions to carry out tasks

  Limitation: **Model fixed** in advance

  How about unknown environment?

- Autonomously acquire model of environment through **learning**

  Limitation: **fixed language** in which learned models can be represented

THIS PAPER

- Robots learns and also
  extends its modelling language

In Machine Learning:

Modelling language = “Hypothesis language”

XPERO PROJECT: ROBOT GAINS “INSIGHTS”

- Goal of XPERO is not only discovering laws that enable predictions, but also *new insights* that make new discoveries easier

- In XPERO: an “insight” in strong sense means something conceptually more *general* and more *abstract* than a law

WHAT IS “INSIGHT”?

INSIGHT ≈

DISCOVERY OF A USEFUL ABSTRACT CONCEPT

WORKING FORMAL DEFINITION = ?

“INSIGHT”

- A definition of insight in the spirit of XPERO:

  an insight is a new piece of knowledge that makes it possible to simplify the current agent’s theory about its environment

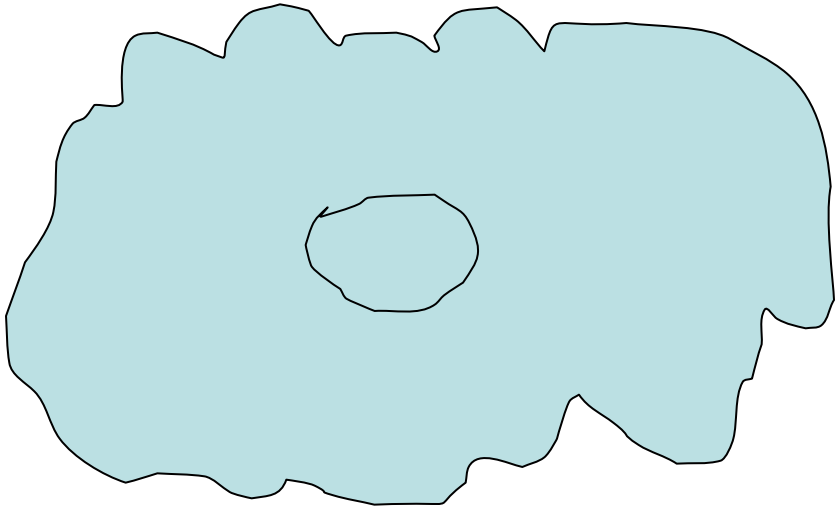
Examples of insights

- Discoveries of notions like:
  - absolute coordinate system,
  - numbers, mathematical theorems (e.g. Pithagora's theorem)
  - notion of gravity
  - notion of support between objects
  - stability
  - notion of force between objects
  - mobility, obstacle, ...
  - functional roles of objects, e.g. a tool

INSIGHTS IN EVOLUTION OF THEORIES

A small initial theory Theory keeps growing ...

Now insight happens, and theory shrinks!



Note: This work is not a typical robotics project

- In a typical robotics project: goal is to improve the robot's performance at some task
- XPERO: improving the robot's theory and "understanding" about the world
- New insights also make further learning easier for the robot (improved representation!)

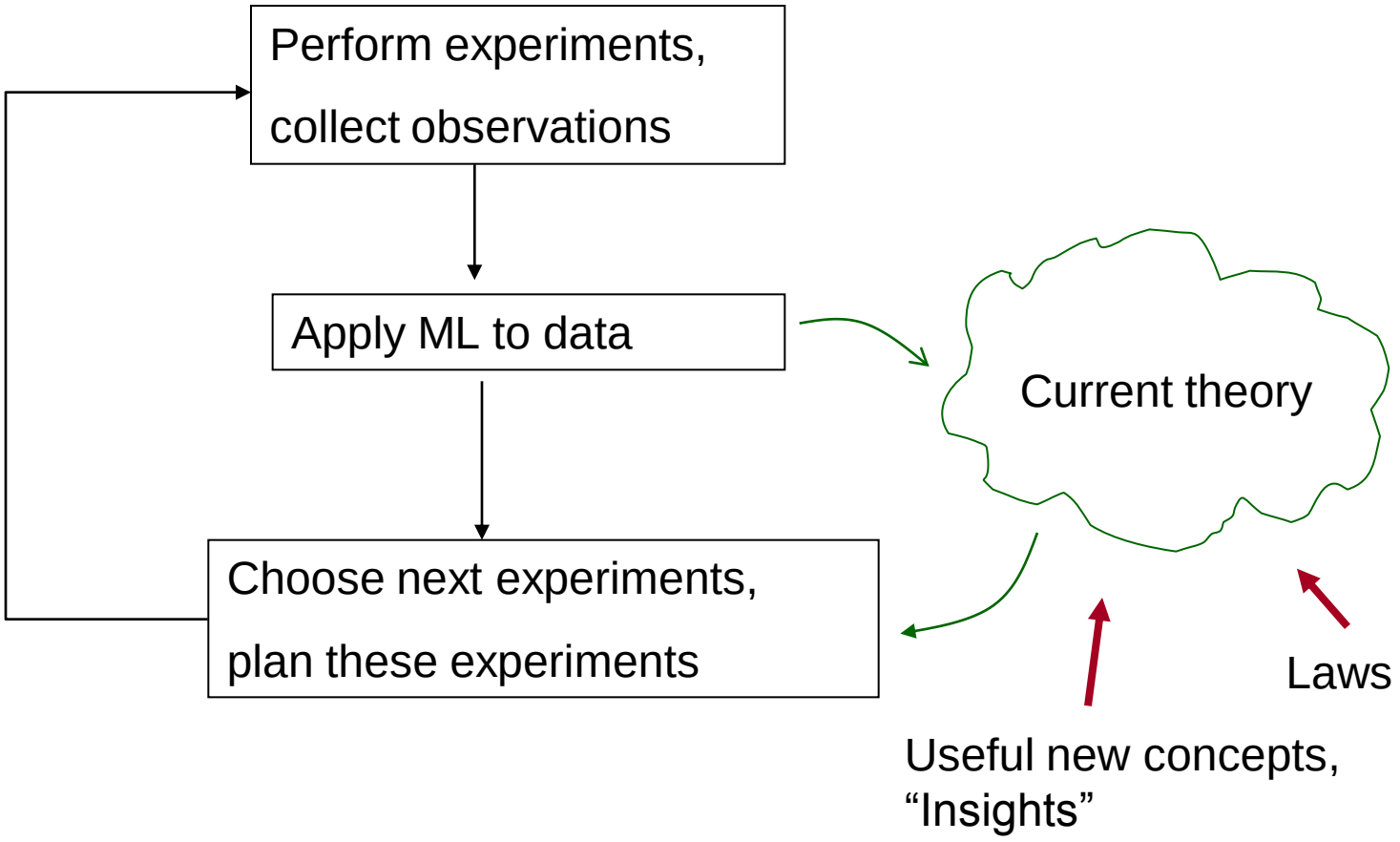
EXPERIMENTAL SCENARIO

- Robot performs experiments in its environment
- Collects observations
- Learns from observations the laws of environment
- Robot also extends its hypothesis language
- That is:

  Discovers new abstract concepts that enable

  easier formulation of new theories about robot's world

Experimental loop



AMBITION

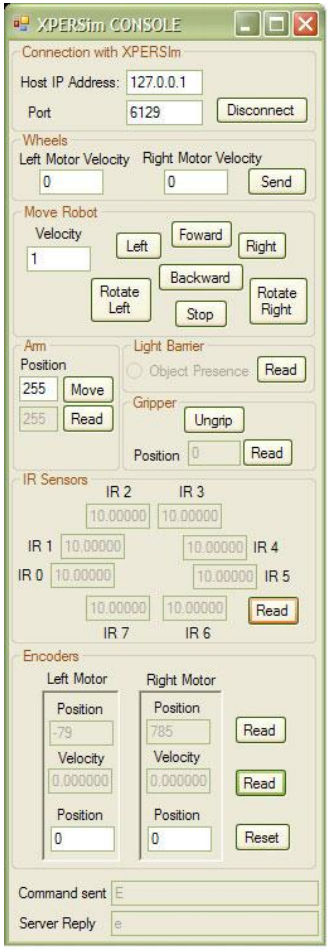
- Find mechanisms that enable the discovery of abstract concepts
- Make only a few rather basic assumptions about the agent's prior knowledge
- Demonstrate how discovery and gaining insights may come about from only a minimal set of "first principles"

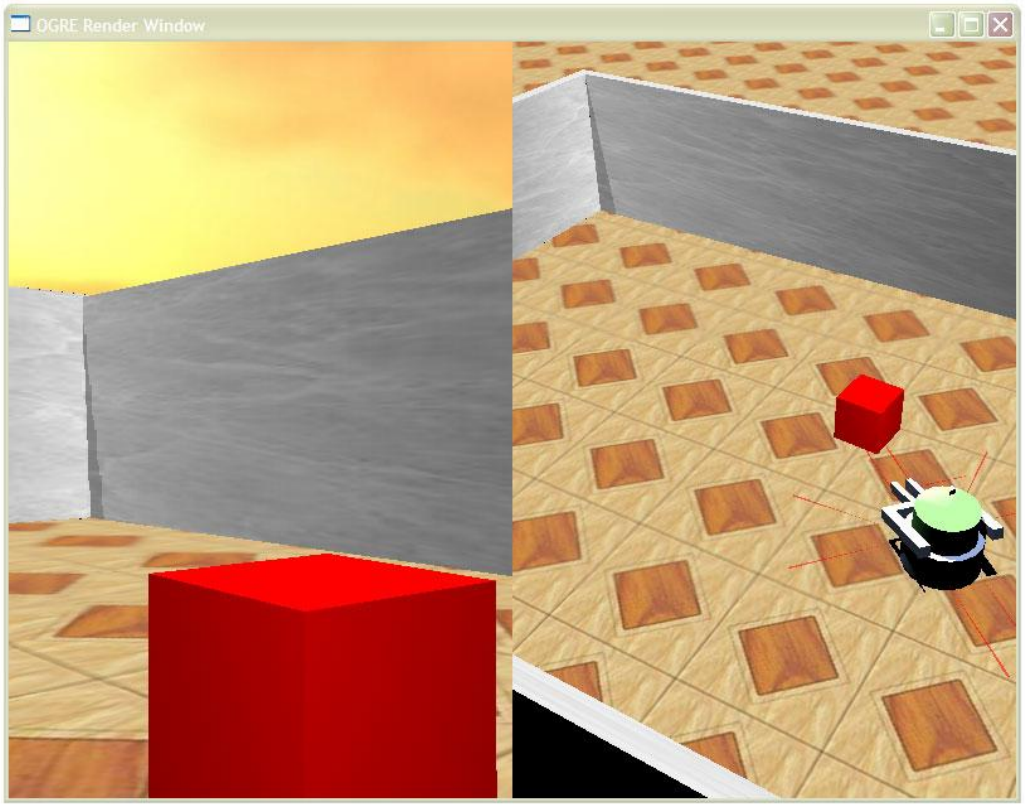
Insights gained by robots in XPERO

- movable and non-movable object,
  - several experimental platforms,
  - single or multiple robots,
  - in simulation and with real robots

- obstacle, several formulations, ...
- movable composite object (train of blocks of any length)
- stability
- degrees of freedom
- tool

Example: red object experimental scenario





Some demos, simulated and real robots

- Simulated Khepera + red ball
- Lego robot + red ball
- Nao robot
  - orientation
  - stability

Machine Learning methods used

- ML methods in Orange and Weka ML platforms
  - decision and regression trees
  - model trees (M5)
  - if-then rules (CN2)

- Other ML methods
  - Learning qualitative trees and rules (QUIN,Pade)
  - Learning (differential) equations (Goldhorn)
  - Inductive Logic Programming (HYPER, Aleph)

ML methods not used

- Insights have to be stated in explicit symbolic form
- This makes some methods unsuitable (neural nets, SVM)

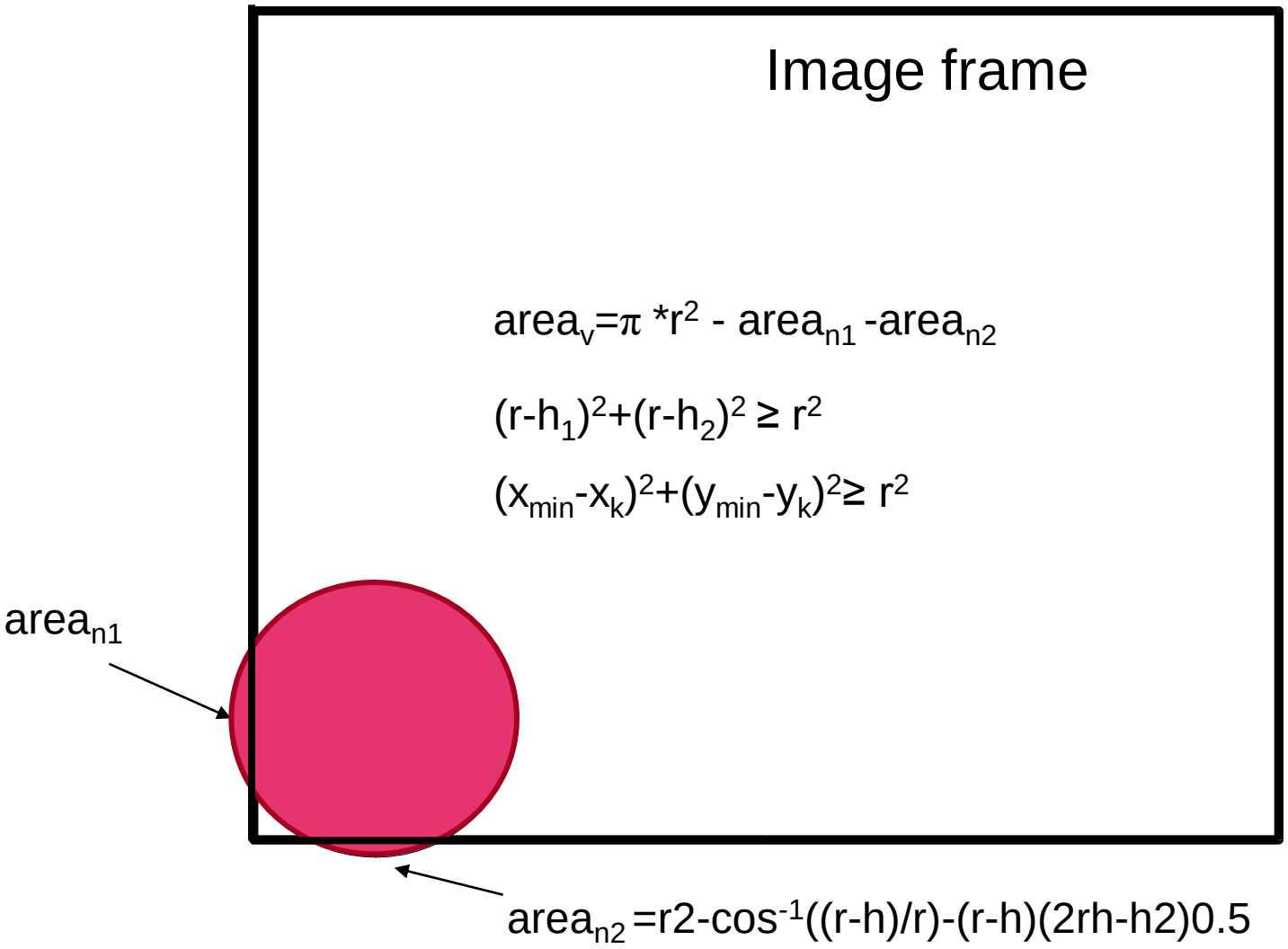
Learning qualitative descriptions

- Instead of quantitative relations, *qualitative* relations may be easier to learn
- For example: When robot is turning away from object, at some point, object area starts decreasing

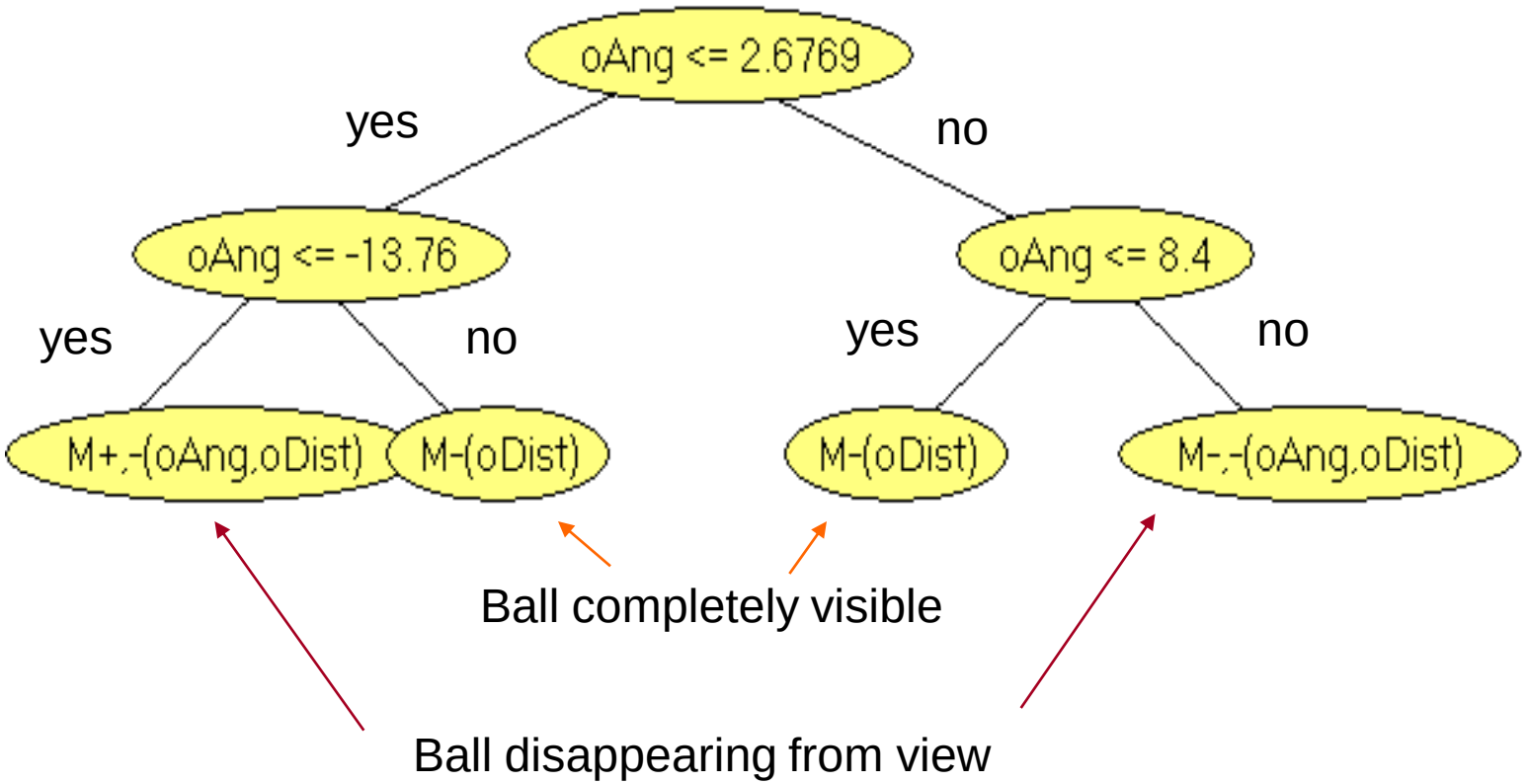
  $\text{Area} = M^{-}(\text{Angle})$

- Therefore we also used qualitative learning methods

Object disappearing from camera view



QUIN's Qualitative Tree for Area



The middle two leaves that apply approximately when the robot is facing the object say that the area decreases with distance. The leftmost and the rightmost leaf describe similar decreasing dependence on object distance, but also describe the disappearing of the object from the camera when the robot is turning away from the object

Comment

- Using numerical learning: accuracy not bad - OK for prediction
- But clarity not promising re. insights, reasoning about domain

ASSESSMENT, CTD.

- The more promising line seems to be the learning of *qualitative models*, as opposed to numerical models

- Possibility, not yet done: Combination of qualitative and quantitative learning (Q^2 learning)

Learning in Logic,

Inductive Logic Programming (ILP)

Abstract insights

ILP PROGRAM USED

- HYPER, HYPothesis refinER
- (Bratko 2001 Prolog for AI book; Leban, Bratko 2009)

ILP ENABLES PREDICATE INVENTION

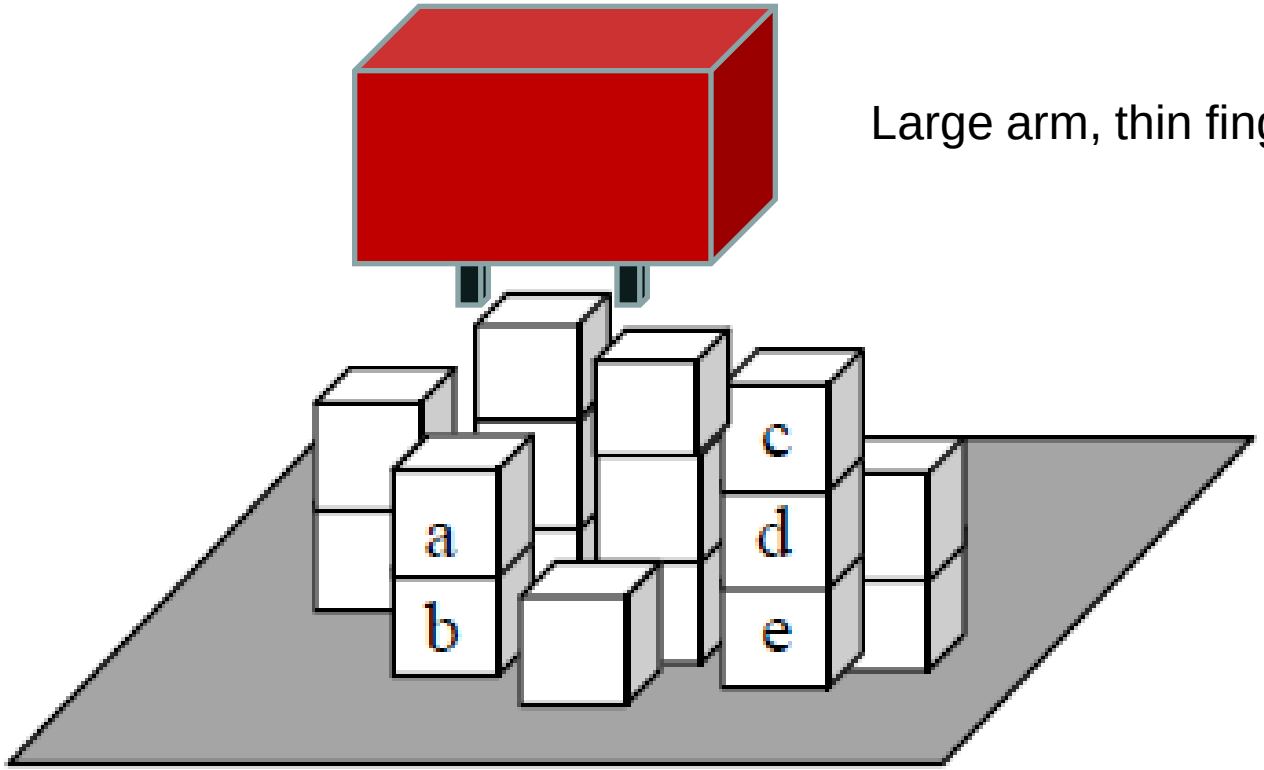
- ILP may invent new predicates during learning
- In this way ILP automatically extends its hypothesis language
- Why is predicate invention important?
- First: to enable easier further learning - new models easier to define with
- Second: sometimes learning is not possible without a new predicate - e.g. learning recursive definitions

RUSSELL AND NORVIG 2010

“Some of the deepest revolutions in science come from the invention of new predicates and functions – for example, Galileo’s invention of acceleration, or Joule’s invention of thermal energy. Once these terms are available, the discovery of new laws becomes (relatively) easy.”

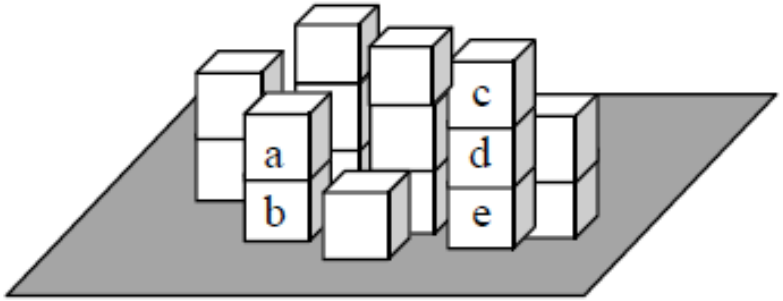
WHY NEW PREDICATES: EXAMPLE

- Robot with large arm manipulating blocks on table



- Block c is graspable, a is not

LEARNING ABOUT GRASPABLE



Robot can see with the camera:

on(a,b),
on(c,d).
on(d,e).
onfloor(b).
onfloor(e).

LEARNING ABOUT GRASPABLE

- Given:

(1) positive and negative examples of graspable blocks:

graspable(c).

not graspable(d).

not graspable(a).

...

(2) “Background knowledge”:

on(a,b).

onfloor(b).

...

- Learn a definition of graspable

LEARNING ABOUT GRASPABLE

- A block B is graspable if there is no other block above B. In logic programming language Prolog, this is stated as:

```
graspable( Block)  :-
    not above( AnotherBlock, Block).
                    % no other block is above Block
```

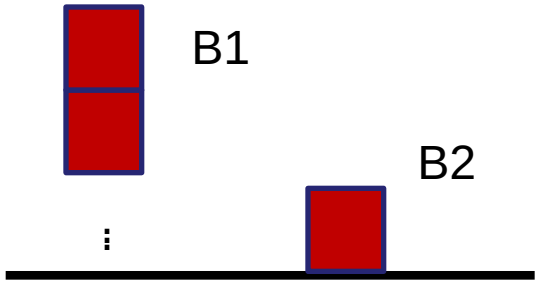
- But predicate above(Block1,Block2) is not given.
- So HYPER has to invent it “from nothing” - just a sa useful auxiliary concept

INDUCED DEFINITION

(1) Block B1 is above block B2 if

B2 is on the floor,

and B1 is on any other block.

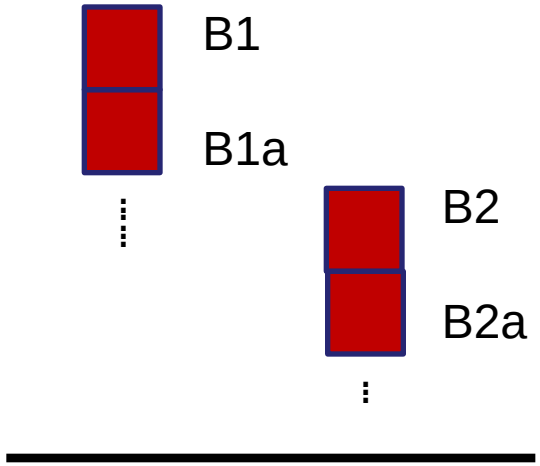


(2) Block B1 is above B2 is if

B1 is on some block B1a,

and B2 is on some block B2a,

and B1a is above B2a.



INDUCED DEF. IN PROLOG

```
above( B1, B2)  :-                  % B1 is above B2 if
    onfloor( B2),                   % B2 is on floor, and
    on( B1, AnotherBlock).          % B1 is on AnotherBlock
```

```
above( B1, B2)  :-                  % B1 is above B2 if
    on( B1, B1a),                   % B1 is on B1a, and
    on( B2, B2a),                   % B2 is on B2a, and
    above( B1a, B2a).               % B1a is above B2a
```

NOTE: This is recursive definition, not possible without invented predicate above(...)

A SCENARIO WHERE INSIGHT HAPPENS THROUGH PREDICATE INVENTION

- Robot is exploring a domain with several objects
- Robot's experiments: execute commands of the form:

  move(Object, Pos, Dist)

- The result of action can be:

  (1) object has moved to a new position (by given distance),
  (2) or object's position has not changed.

- Robot has collected experimental data of form

at(Obj, T, P)

Obj was observed at position P at time T

- Robot wants to learn to predict result of action, i.e.:

move(Obj, P1, D, P2)

command "move Obj from P1 by distance D" resulted in Obj at P2

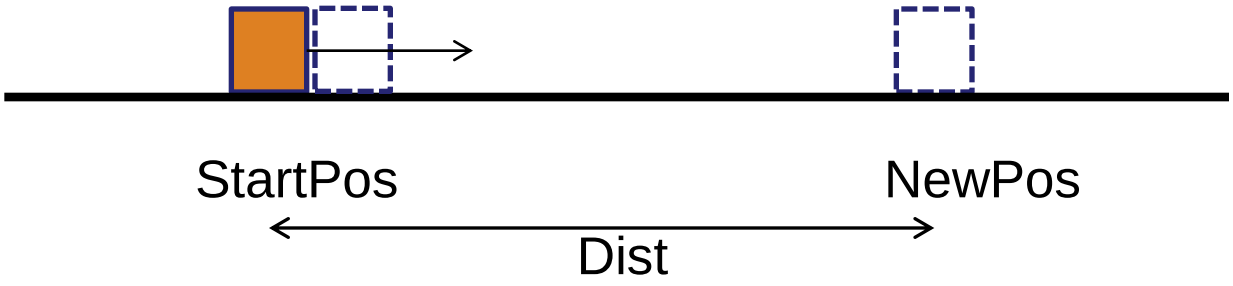
Robot’s prior knowledge

different(X, Y)

vectors X and Y are different

add(X, Y, Z)

Z = X + Y , vector summation



- An easy way to predict about moving an object O would be:

  If O is movable then NewPos = StartPos+Dist,

  else NewPos = StartPos

- Note: Robot has no idea of movable or not movable object; concept “movable” was not mentioned in the problem statement at all

- To learn to predict results of actions, **robot invented new notion of “movable object”**

Negative examples

- Nature only provides positive examples - only what may really happen in nature
- ILP also needs negative examples - how can these be obtained? (rather common problem of ILP)
- Constructed through a kind of closed-world assumption; NewPos is a *function* of Obj, Dist, NewPos:

    if ex(move(O,P1,D,P2)) then

    forall(P3): if P3 =\= P2 then nex(move(O, P1, D, P3)

An induced theory about object pushing

```
p( Object):-                    % Invented p, p means "movable"
    at( Object, Time1,Location1),
    at( Object, Time2, Location2),
    different( Location1, Location2).   % Location1 =\= Location2

move( Object,C,B,C):-
    not p( Object).

move( Object,B,C,D):-
    add(C,B,D),
    p( Object).
```

Insight: notion of movable object!

```
p(Obj):-
    at(Obj,T1,P1),
    at(Obj,T2, P2),
    different( P1, P2).
```

The invented predicate corresponds to discovering the notion of movable object!

To the robot, movable object is an object that has been observed at different places in the past.

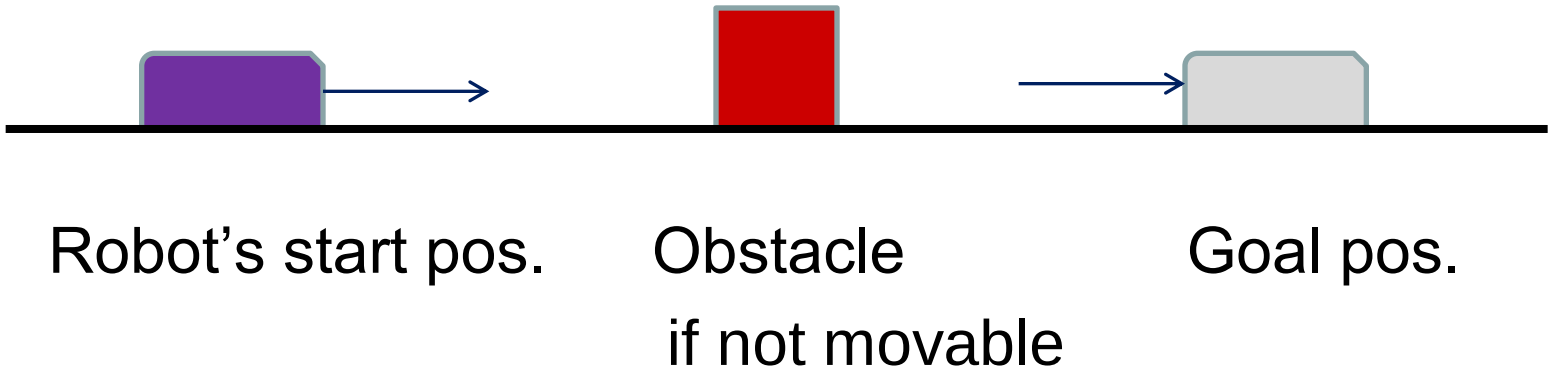
Relative movability

- In another experiment: Two robots, a stronger and a weaker one:

  HYPER discovered that some objects can be moved by any robot, and some by the stronger robot only

Discovery of concept of “obstacle”

- In another experiment HYPER gained insights of **obstacle** as an immobile object between robot’s current position and robot’s goal position

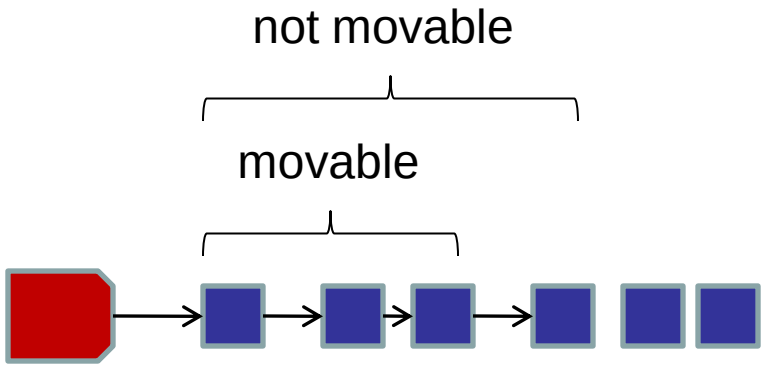


Learning about pushing a “train” of blocks

```
p(Objs):-        % objs movable
    ex(moveTrain(Objs2, To, StartTime, End)),
    approxEqual(To, End),
    heavier(Objs2, Objs).

moveTrain(Objs, To, StartTime, End):-
    approxEqual(To, End),
    p(Objs).

moveTrain(Objs, To, StartTime, End):-
    itemSubset(Objs, ObjSubset),
    last(Rest, Obj, ObjSubset),
    not p(ObjSubset),
    p(Rest),
    approxAt(Obj, StartTime, End).
```



Learning about functions of objects

- Function of an object is not a property of an object – the function is determined by the way we use an object
- Instead of individual actions we have to analyze *plans* executed by the robot
- Example of a function:
  - **Concept of a tool** – a tool is an object that the agent can use to achieve certain goal

Plans as sequences of pairs *Action → Goals*

- Knowing only the sequence of actions is not sufficient to determine the function of an object
- We also need information about the intended goals of individual actions
- To find these goals we had to combine our planning algorithm with a means-ends planner
- An example of a final plan:

move([0.0,0.0], [4.0,0.0]) → [at(robot, [4.0,0.0])]
push(block1, [4.0,4.0], [4.0,6.0]) → [at(block1, [4.0,6.0]), at(robot, [4.0,4.9])]
move([4.0,4.9], [_X1,4.9]) → [at(robot, [_X1,4.9])]
move([_X1,4.9], [_X1,6.0]) → [at(robot, [_X1,6.0])]
push(block1,[4.0,6.0], [14.0,6.0]) → [at(block2, [15.0,6.0])]
{_ X1 < 3.0}

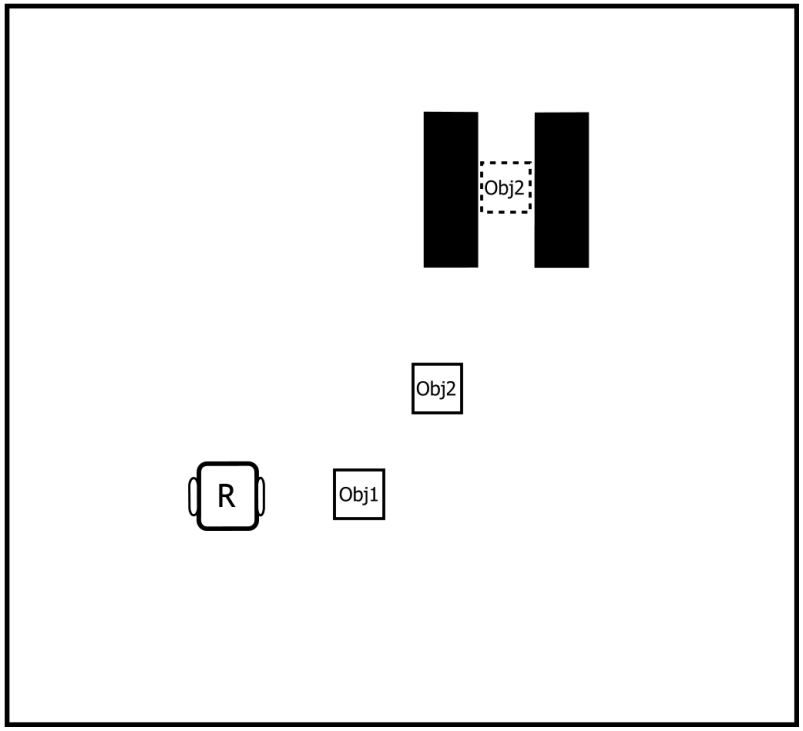
Unspervised learning

- We don’t have information whether a plan uses a tool or not
- We can search for repeating patterns (subplans) in the plans
- Measures for determining the interestingness of a pattern:
  - Number of repetitions of a pattern = *Count*
  - Length of the pattern = *Len*
- We compute the score of a pattern based on the principles from data compression:

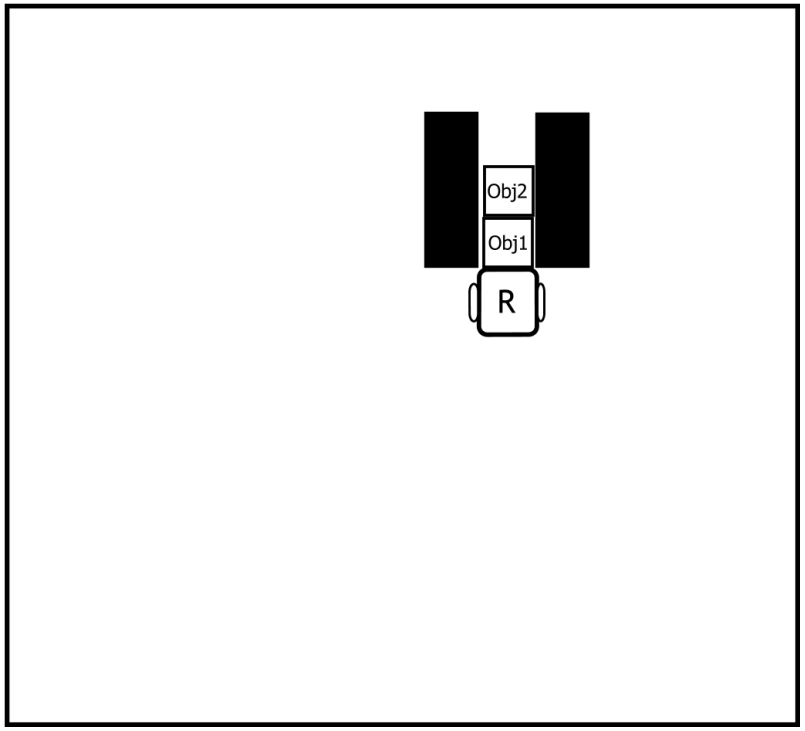
  *Score = Count * (Len-1) - Len*

Score= the saving in encoding length of all plans if we introduce the pattern as a macro operator

An example of tool use



START STATE

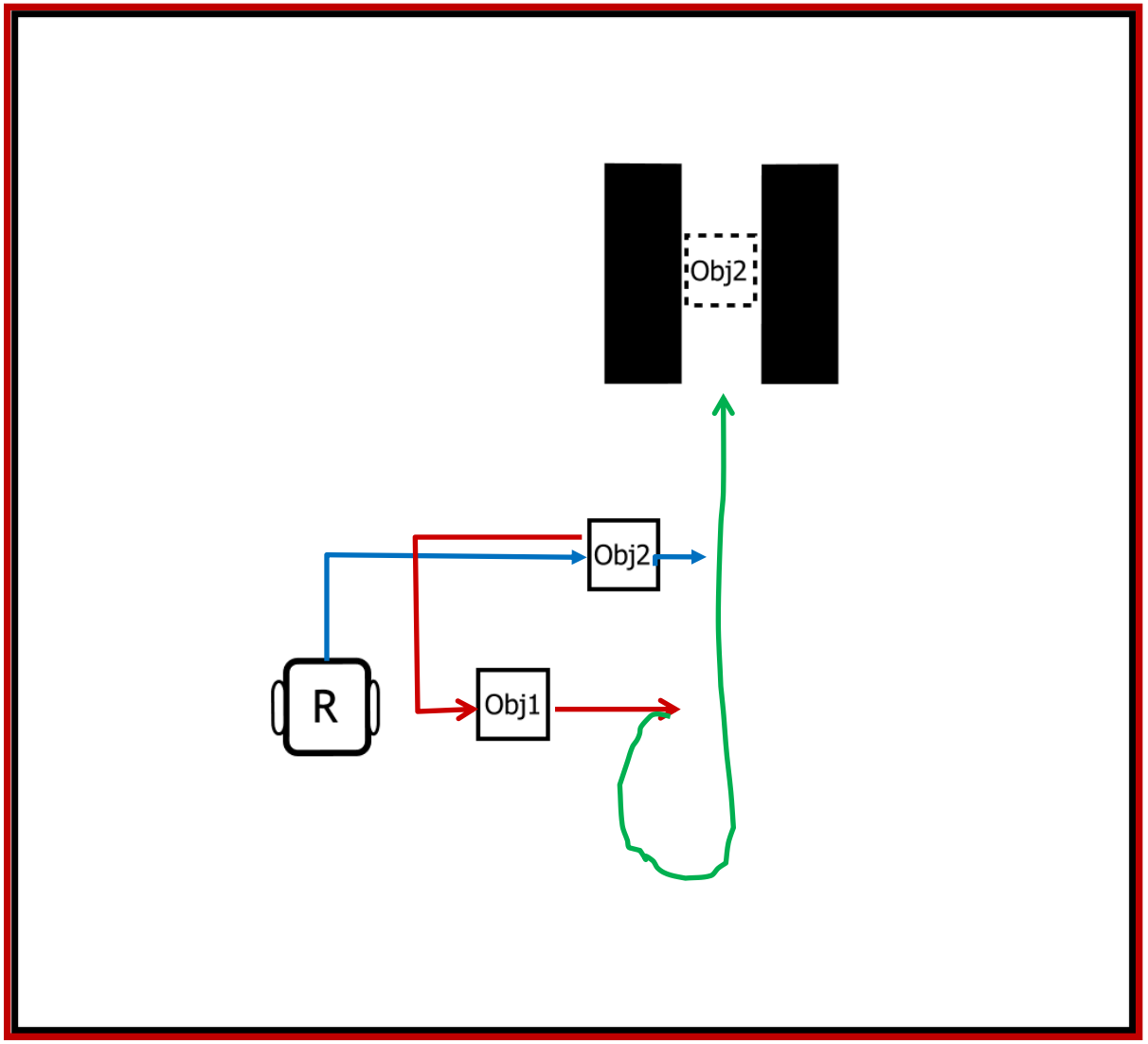


END STATE

- Object *Obj1* is an example of a tool, since it is used to move object *Obj2* to the goal position

A plan to achieve the goal

START STATE



FINDING USEFUL PATTERNS IN PLANS

- Search for repeating patterns (subplans) in the plans
- Select patterns that maximise data compression score:

$$Score = Count * (Len-1) - Len$$

- *Score*= the saving in encoding length of all plans if we introduce the pattern as a macro operator

Learning the tool concept using Hyper

- Analysis of the 10 best-scored patterns found using the unsupervised learning
- Use Hyper to find a description which separates *a* pattern from other nine patterns

- Background knowledge:

  appears(Object, Expr): *Object appears in Expr*

  member(Action→ Goals, Plan): *Action, Goals in Plan*

  object(X): *X is an object*

Learned descriptions of best patterns

- Hyper found a description for three patterns:

```
pattern_2( Patt)   :–
    length(Patt, 4),
    member( Action → Goals,  Patt),
    appears( Obj, Action),
    not appears( Obj, Goals),
    object( Obj).

pattern_3( Patt)   :–
    length(Patt, 5),
    member(Action → Goals, Patt),
    appears( Obj, Action),
    not appears( Obj, Goals),
    object( Obj).
```

```
pattern_6( Patt)   :–
    length(Patt, 3),
    member( Action → Goals, Patt),
    appears( Obj, Action),
    not appears( Obj, Goals),
    object( Obj).
```

LEARNING TO CHARACTERISE BEST PATTERNS

- If we allow HYPER to invent a new predicate,
  can HYPER find a shorter theory ?

- Yes!

- The invented predicate can be interpreted as an abstract notion of a tool

LEARNED THEORY WITH INVENTED PREDICATE P

```
p(Obj, Patt):-
    object(Obj),
    member(Action → Goals, Patt),
    appears(Obj, Action),
    not appears(Obj, Goals).
```

```
pattern_3(Patt):-
    length(Patt, 5),
    appears(Obj, Patt),
    p(Obj, Patt).
```

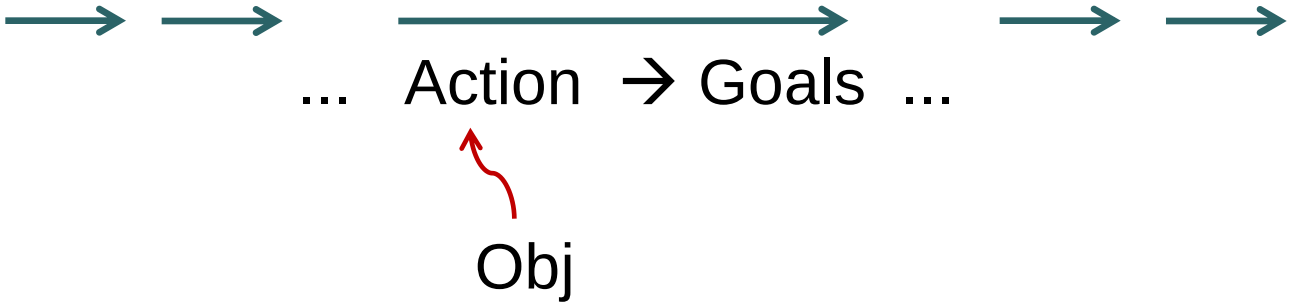
```
pattern_2(Patt):-
    length(Patt, 4),
    appears(Obj, Patt),
    p(Obj, Patt).
```

```
pattern_6(Patt):-
    length(Patt, 2),
    appears(Obj, Patt),
    p(Obj, Patt).
```

INVENTED PREDICATE IS A NOTION OF A TOOL

```
tool(Obj, Plan):-      % Obj has the role of a tool in Plan
    object(Obj),
    member(Action → Goals, Plan),
    appears(Obj, Action),
    not appears(Obj, Goals).
```

Plan



Conclusions

- Robot “gains insights”, i.e. discovers abstract new concepts
- ML approach to discovering abstract concepts: predicate invention in Inductive Logic Programming
- Discovered abstract concepts:
- movable, obstacle, “train”, tool, ...
- Another useful feature: Use qualitative representations
- Demo: Learning and planning with qualitative representation to manipulate objects by pushing (Troha)

ACKNOWLEDGEMENTS

- European project XPERO

- ARRS, Slovenian Research Agency

- Members of Ljubljana team include:
- Jure Žabkar, Gregor Leban, Janez Demšar, Aljaž Košmerlj, Tadej Janež, Miha Troha, Sašo Moškon, Simon Kozina, ...