CHALMERS

Artificial Intelligence – TIN172

Project presentation

Group 20

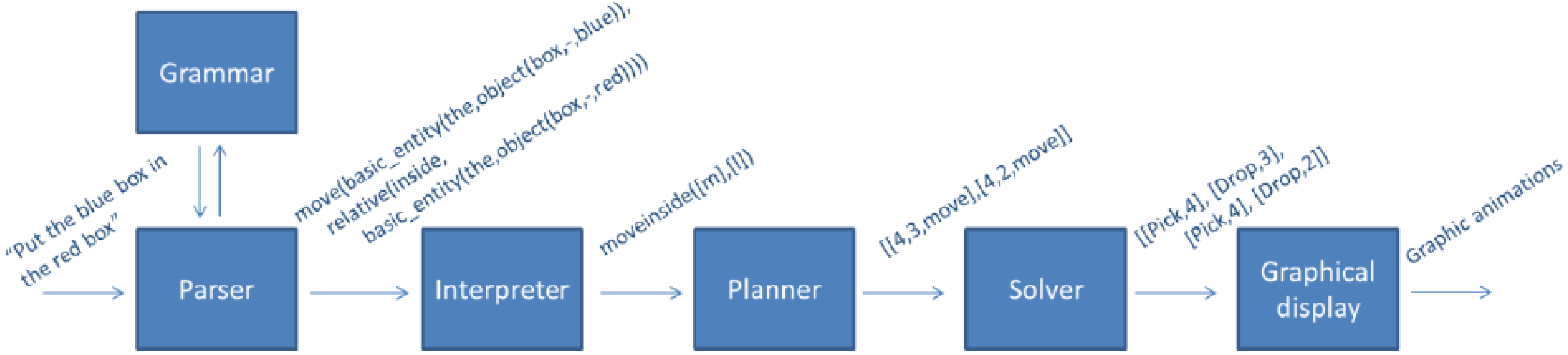
Pierre BOUTRY, Dan DOLONIUS, Julien MICHELET, Emeric ROVERC'H

21/05/2014

Agenda

1. Presentation of the application
2. Grammar
3. Interpreter
4. Planner
5. Ambiguities handling
6. Possible improvements

Presentation of the application


Grammar
"Put the blue box in the red box"
Parser
move(basic_entity(the,object(box,-,blue)), relative(inside, basic_entity(the,object(box,-,red))))
Interpreter
moveinside([m],[l])
Planner
[[4,3,move],[4,2,move]]
Solver
[[Pick,4], [Drop,3], [Pick,4], [Drop,2]]
Graphical display
Graphic animations

- Simple client-server architecture
- On client side: Collect of the query and graphical display
- On server side: Parser, Interpreter, Planner, Solver
- We used **Prolog** for the project

Agenda

1. Presentation of the application

▶ 2. Grammar

3. Interpreter

4. Planner

5. Ambiguities handling

6. Possible improvements

Grammar

- One of the major improvements is the possibility for the user to ask **3 new types of questions** containing the following keywords: where, what and count.
- For example:
  - *“where is the white big ball?”*
  - *“what are the objects in the world?”*
  - *“how many ball are in the world?”*

- The user can also perform request to obtain information about an entire **stack**.
  - *“what are the objects in stack 2?”*

Agenda

1. Presentation of the application

2. Grammar

▶ 3. Interpreter

3. 1 Interpretation rules

3. 2 Quantifiers

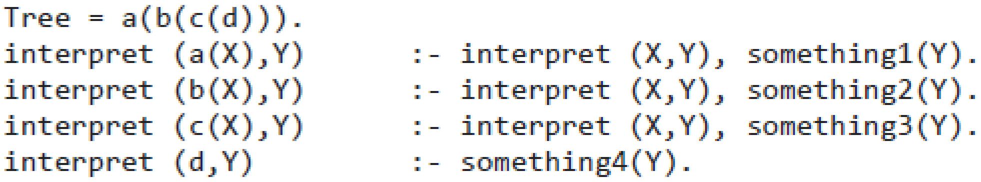
4. Planner

5. Ambiguities handling

6. Possible improvements

Interpreter – Interpretation rules (1/2)

- We make a set of rules to **recursively** satisfy every node in the tree, e.g:

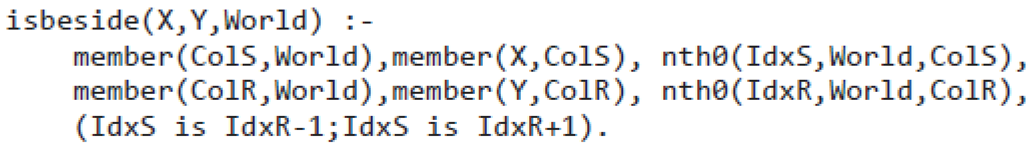


```
Tree = a(b(c(d))).
interpret (a(X),Y)      :- interpret (X,Y), something1(Y).
interpret (b(X),Y)      :- interpret (X,Y), something2(Y).
interpret (c(X),Y)      :- interpret (X,Y), something3(Y).
interpret (d,Y)         :- something4(Y).
```

- We implemented a set of **62 rules** to:
  - *Get object satisfying description.*
  - *Handle **the**, **any**, or **all** cases for basic entities.*
  - *Get object which satisfies relation to "object X".*
  - *Get object which satisfies relation to stack.*
  - *Process output to goal.*

Interpreter – Interpretation rules (2/2)

- Further, we have some rules to see if an object is e.g. beside any other object



```
isbeside(X,Y,World) :-
    member(ColS,World),member(X,ColS), nth0(IdxS,World,ColS),
    member(ColR,World),member(Y,ColR), nth0(IdxR,World,ColR),
    (IdxS is IdxR-1;IdxS is IdxR+1).
```

- We also added rules to support the new count, where and what questions.

Agenda

1. Presentation of the application

2. Grammar

▶ 3. Interpreter

   3. 1 Interpretation rules

   3. 2 Quantifiers

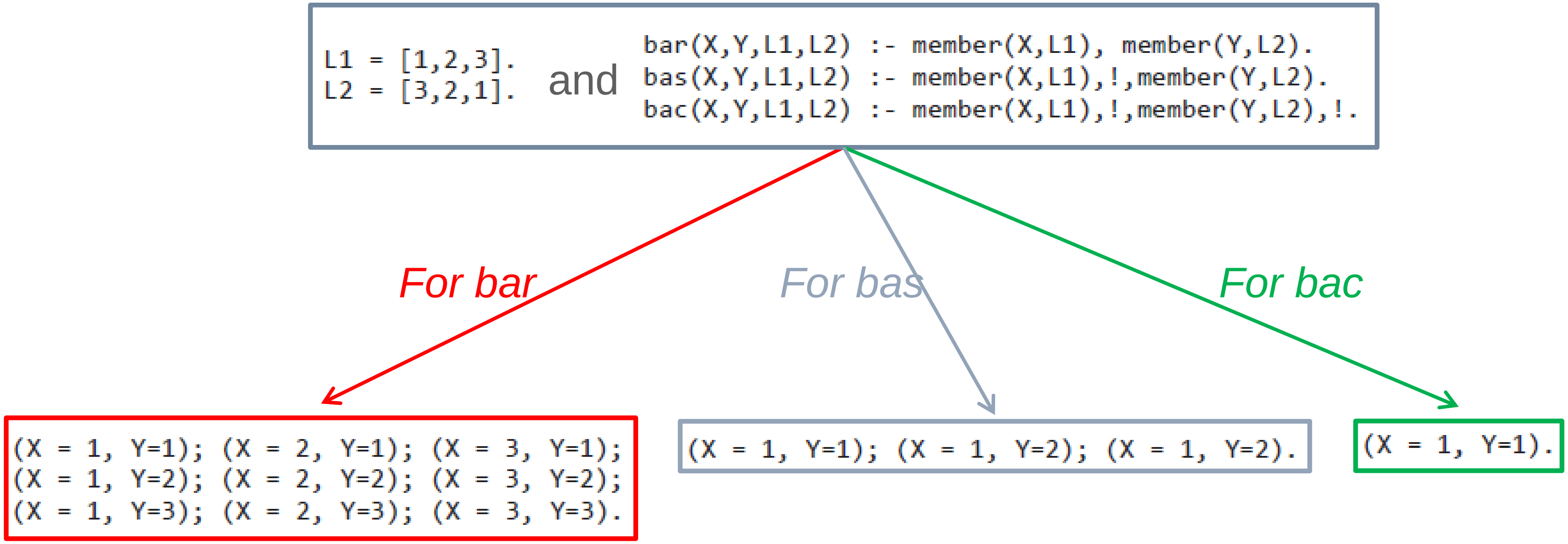
4. Planner

5. Ambiguities handling

6. Possible improvements

Interpreter - Quantifiers

- The quantifiers allow the robot to handle query such has
  - *“put any ball in the red box”.*
- The quantifier function uses cuts to choose one possible action
- A cut (!) in prolog is a way to restrict the search by “fixing” the variables
- Here is an example, we define:



```
L1 = [1,2,3].
L2 = [3,2,1].
```

and

```
bar(X,Y,L1,L2) :- member(X,L1), member(Y,L2).
bas(X,Y,L1,L2) :- member(X,L1),!,member(Y,L2).
bac(X,Y,L1,L2) :- member(X,L1),!,member(Y,L2),!.
```

For bar

```
(X = 1, Y=1); (X = 2, Y=1); (X = 3, Y=1);
(X = 1, Y=2); (X = 2, Y=2); (X = 3, Y=2);
(X = 1, Y=3); (X = 2, Y=3); (X = 3, Y=3).
```

For bas

```
(X = 1, Y=1); (X = 1, Y=2); (X = 1, Y=2).
```

For bac

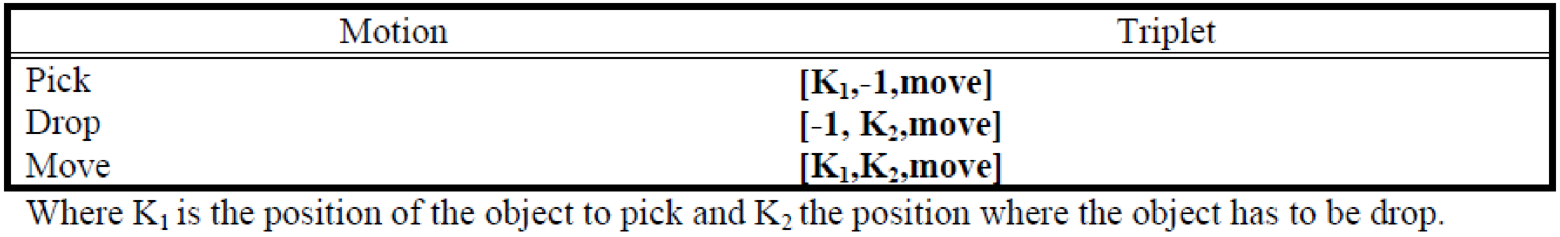
```
(X = 1, Y=1).
```

Agenda

1. Presentation of the application
2. Grammar
3. Interpreter
4. ▶ Planner
5. Ambiguities handling
6. Possible improvements

Planner

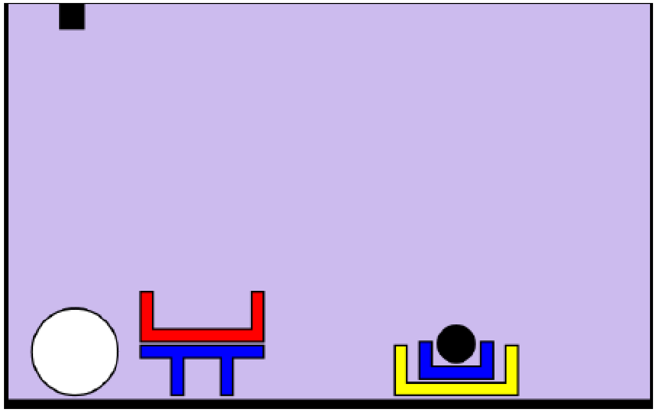
- The planner takes a query as an input and build **list of triplets**

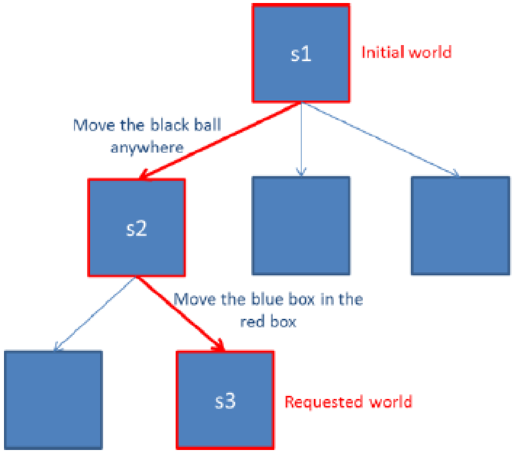


Motion	Triplet
Pick	**[K_1,-1,move]**
Drop	**[-1, K_2,move]**
Move	**[K_1,K_2,move]**

Where K_1 is the position of the object to pick and K_2 the position where the object has to be drop.

- Here is an example for a complex case:
  - *“Move the black ball in the red box”.*

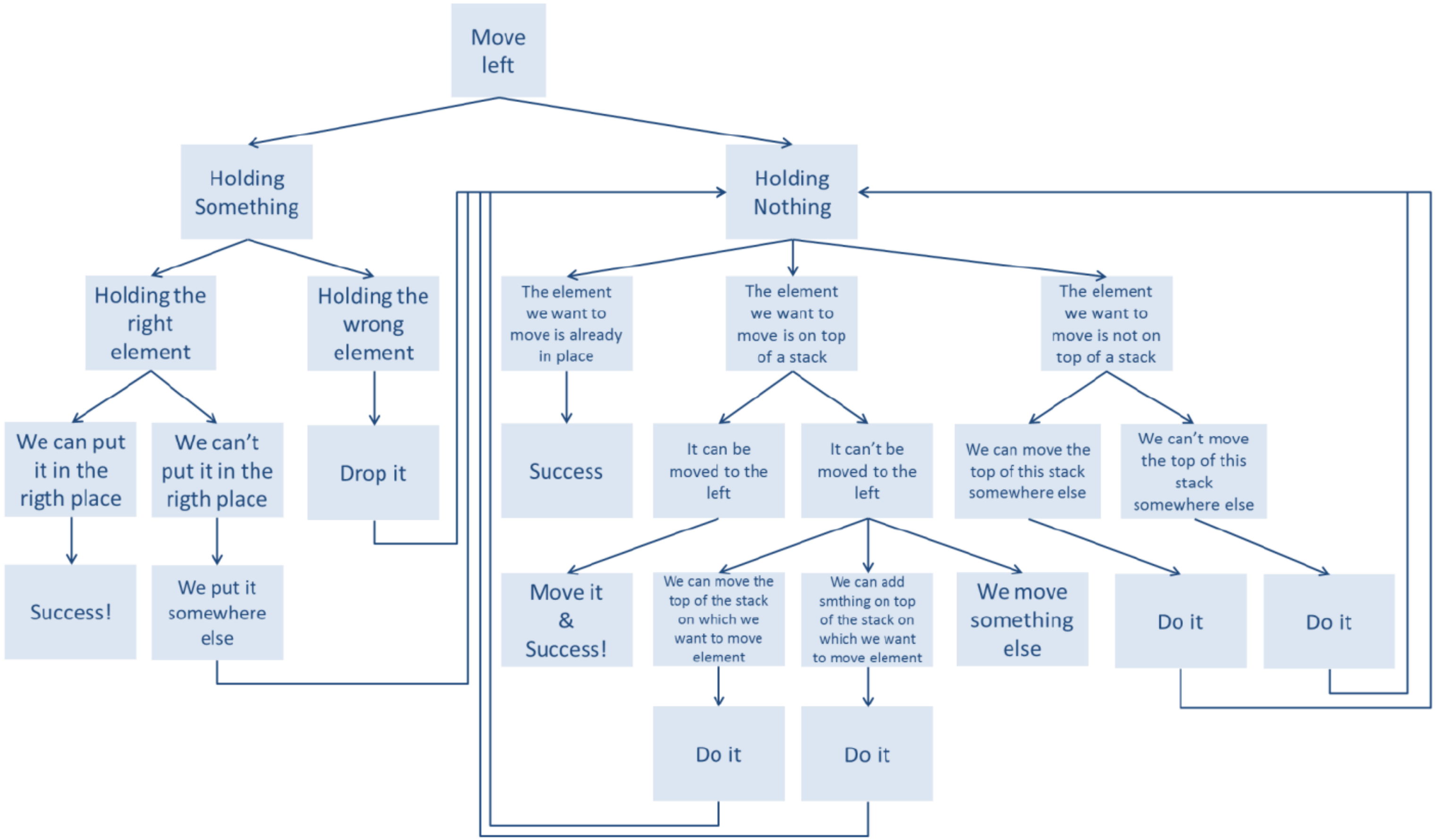



s1
Initial world
Move the black ball anywhere
s2
Move the blue box in the red box
s3
Requested world

- And the output of the planner is:
  - *“Plan: pick,2 drop,1” where 2 is the position of the black ball (Stack 2) and 1 the position of the red box (Stack 1)*

Planner - Heuristic

- The heuristic allows the robot to handle complex cases in a **smarter** way
- For example, in case of a *move left* query:



Move left

Holding Something

Holding Nothing

Holding the right element

Holding the wrong element

We can put it in the rigth place

We can't put it in the rigth place

Drop it

Success!

We put it somewhere else

The element we want to move is already in place

The element we want to move is on top of a stack

The element we want to move is not on top of a stack

Success

It can be moved to the left

It can't be moved to the left

We can move the top of this stack somewhere else

We can't move the top of this stack somewhere else

Move it & Success!

We can move the top of the stack on which we want to move element

We can add smthing on top of the stack on which we want to move element

We move something else

Do it

Do it

Do it

Do it

Agenda

1. Presentation of the application

2. Grammar

3. Interpreter

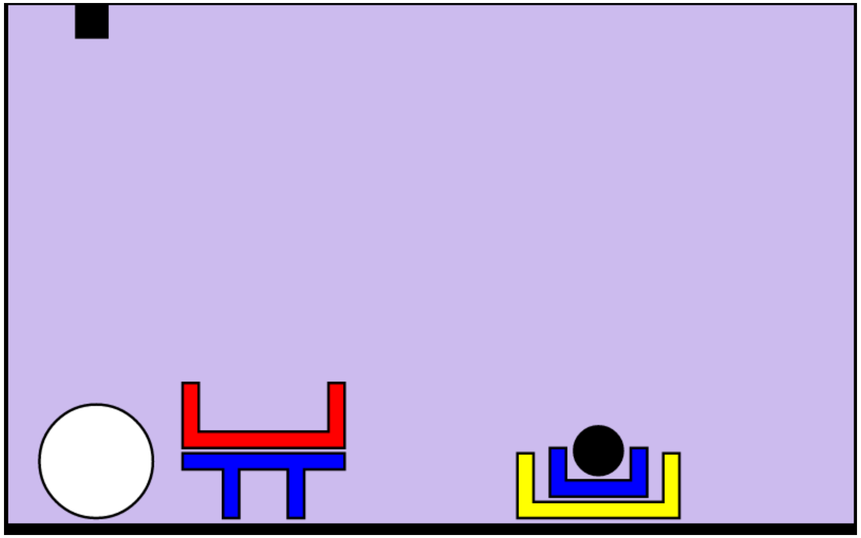
4. Planner

▶ 5. Ambiguities handling

6. Possible improvements

Ambiguities handling

- Allows the robot to **ask question** if a **query is not clear** enough
- For example, in the world below:
  - *“what is under the box?”*



- The machine would ask for more information
  - *“What object do you mean?”*
- The user will then have to specify his query
  - *“The big red one”*

- We had to use a file to store the information needed and reload the prolog part

Agenda

1. Presentation of the application

2. Grammar

3. Interpreter

4. Planner

5. Ambiguities handling

▶ 6. Possible improvements

Possible improvements

- Use machine learning for the heurisitc
- Use of probabilistic planning for the heuristic
- Use SAT (Boolean Satisfiability problem) to solve our problem

Questions