

Concurrence et Répartition

Inspector Prolog

Joao Costa Da Quinta, Lea Heiniger

16.05.2023

Inspector prolog

- Educational games to learn formal logic
- Implemented in prolog
- Cases to investigate

Lying suspects

the possibility that the suspects may be lying makes the scenarios more interesting

```
tells_the_truth(P) :- pretended_location(P, L),  
                        was_in(P, L).  
  
liar(P) :- \+tells_the_truth(P).
```

→ If a suspect was somewhere other than where he pretended to be then he is lying.

Lying suspects

A liar can be innocent of the crime but the information he gives cannot be trusted !

```
was_in(P, L) :-person(P), forall(location(X), X=L).  
was_in(P, L) :-lost(P, O), found_in(O, L).
```

```
% Old rule :
```

```
% was_in(S, L) :-saw(P, S), not_guilty(P),  
% was_in(P, L).
```

```
% Updated to :
```

```
was_in(S, L) :-saw(P, S), tells_the_truth(P),  
was_in(P, L).
```

Scenario 2

The scenario is implemented using the lying suspects mechanics and is working correctly.

```
...  
pretended_location(charles_hatant, boudoir).  
pretended_location(fab_ullateur, boudoir).  
saw(fab_ullateur, charles_hatant).  
...  
?- was_in(fab_ullateur, L).  
true;  
L = chambre_2.  
?- liar(fab_ullateur).  
true.  
?- has_alibi(charles_hatant).  
false.
```

Details for user

The user will be able to discover information by **examining** locations and clues.

```
?- found_in(0, salle_restaurant).  
0 = collier_s ;  
0 = serviette.  
  
?- has_detail(collier_s, D).  
D = 'le pendentif est en forme de S'.  
  
?- was_in(sally_riez, salle_restaurant).  
true.
```

User interactions : Recall

The user has to be able to :

- Integration of Prolog with Python
- Intuitive User Interface
- Flexible Database

Smokers

Smoking suspects may leave cigarette butts behind them

```
could_be_in(P, L) :-smoker(P), found_in(M, L),  
                    megot(M).
```

```
% Exemple
```

```
person(jean).
```

```
location(room).
```

```
smoker(jean).
```

```
megot(m)
```

```
found_in(m, room).
```

```
?- could_be_in(jean, room).
```

```
true.
```


How to link `could_be_in` to `was_in` ?

First idea :

- Cigarette butts can be attributed to smokers as they are proven to have been in a room.
- If there is an unattributed cigarette butt and a smoker without an alibi then they can be linked together.

Problems :

- What if we have several unattributed butts left in different locations?
- Or several smokers without alibi ?
- ...

Possible solution : Each smoker lives **exactly one** cigarette butt

User interactions : Example

Goal: To create Interactive Query Execution
How to impose the order of queries made by the user ?

- (1) amy_khale lost boucle_oreille_a
- (2) boucle_oreille_a found_in salon
- (3) amy_khale was_in salon

Our next goals

- Fact database
 - ▶ Implement proof system
 - ▶ Scenario 3 writing and implementation
- Implement Interactive Query Execution

Semester plan

	1	2	3	4	5	6	7	8	9	10	11	12	13
refresh prolog knowledge													
choose game mechanics													
design facts database													
create scenarios													
implement facts database													
implement logic													
implement game --> interface													
generative model													

Concurrence et Répartition

Inspector Prolog

Joao Costa Da Quinta, Lea Heiniger

16.05.2023