

# Concepts of programming languages

## Embedding Prolog in C#

Winand, Roald, Sjoerd, Alexey and Anvar



# Contents

- ▶ Problem definition 0.5 min
- ▶ Methodology 0.5 min
- ▶ Implementation 6 min
- ▶ Results 2 min
- ▶ Reflection 2 min



# Problem definition

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Methodology

- ▶ load in terms and clauses
- ▶ implement unification
- ▶ implement backtracking, which will be a milestone
- ▶ implement mathematics
- ▶ implement native Prolog functions such as findall
- ▶ implement negation and cut



# Implementation (1)

Knowledge database

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Implementation (2)

Validation

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Implementation (3)

Representation

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Implementation (4)

Querying

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]



# Implementation (5)

Proof search(1)

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Implementation (6)

Proof search(2)

Placeholder



Universiteit Utrecht

[Faculty of Science  
Information and Computing  
Sciences]

# Results (1)

- ▶ Query: "male(dicky)" Result: True. Expected result: True.
- ▶ Query: "male(*in the KB*)" Result: True. Expected result: True.
- ▶ Query: "male(andrew)" Result: False. Expected result: False.
- ▶ Query: "parent(elmer, don)." Result: True. Expected result: True.



## Results (2)

- ▶ Query: "female(X)." Result: X = anne. X = rosie. X = esther. X = mildred. X = greatgramma. X = mia. X = god. Expected result: same
- ▶ Query: "parent(X, anne)." Result: X = don. X = rosie. Expected result: same.
- ▶ Query: "brother(X, Y)." Result: X = dicky, Y = rosie. X = y = dicky. X = y = randy. X = randy, Y = mike. X = randy, Y = anne. X = Y = randy X = randy, Y = mike. X = randy, Y = anne. X = mike, Y = randy. X = randy, Y = anne. X = Y = mike. X = mike, Y = anne. X = Y = don. X = Y = don. X = Y = blair. X = Y = mel. X = Y = teo. Expected result: same.

*brother(X,Y) :-*

*male(X),parent(Somebody,X),parent(Somebody,Y).*



## Results (3)

- ▶ Query: “uncle(X, teo).” Result: XX = dicky. X = dicky. X = randy. X = randy. X = randy. X = mike. X = mike. X = mike. X = don. X = mel. Expected result: false.

Multiple internal variables not unified correctly:

*uncle(X,Y) :- brother(X,Par),parent(Par,Y).*

- ▶ Query: “american(anne)” Result: True. Expected result: True.

*american(anne).*



## Results (3)

- ▶ Query: `american(X)`. Result: `StackOverflowException`.  
Expected result: *long list of X-Y-assignments*.

Infinite recursion:

*`american(X) :- ancestor(anne,X).`*

*`ancestor(X,Y) :-  
parent(X,Somebody),ancestor(Somebody,Y).`*



# Reflection (1)

- ▶ Working implementation.
- ▶ Nontrivial queries.
- ▶ Actually useful.



## Reflection (2)

- ▶ Large group, sequential project in nature.
- ▶ Unification and backtracking much more complicated than expected with lots of edge cases.

