

# Concepts of programming languages

## Prolog

Winand, Roald, Sjoerd, Alexey and Anvar



# Proof Search

- ▶ The manner in which a query is handled
- ▶ Knowledge Database is read from top to bottom
- ▶ Tries to unify with facts and heads of rules
- ▶ At first valid encounter, unification is carried out
- ▶ Variables are replaced by internal variables (e.g. \_G2145)
- ▶ A search is done in a depth first fashion in a tree-shaped structure



# Backtracking

- ▶ When a search path is not valid, **backtracking** occurs:  
Traversing the tree in opposite direction until a variable binding (choise point) is reached
- ▶ If a result is found, one can choose to continue the search by using backtracking, using the ; command



# A simple example (1)

Knowledge database:

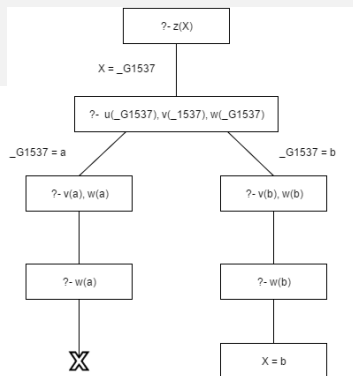
```
u(a) .  
u(b) .  
v(a) .  
v(b) .  
w(b) .  
z(X) :- u(X) , v(X) , w(X) .
```

Figure 1: Knowledge database



# A simple example (2)

```
[trace] 6 ?- z(X).  
Call: (7) z(_G1537) ? creep  
Call: (8) u(_G1537) ? creep  
Exit: (8) u(a) ? creep  
Call: (8) v(a) ? creep  
Exit: (8) v(a) ? creep  
Call: (8) w(a) ? creep  
Fail: (8) w(a) ? creep  
Redo: (8) u(_G1537) ? creep  
Exit: (8) u(b) ? creep  
Call: (8) v(b) ? creep  
Exit: (8) v(b) ? creep  
Call: (8) w(b) ? creep  
Exit: (8) w(b) ? creep  
Exit: (7) z(b) ? creep  
X = b.
```



# A more complicated example (1)

Knowledge database:

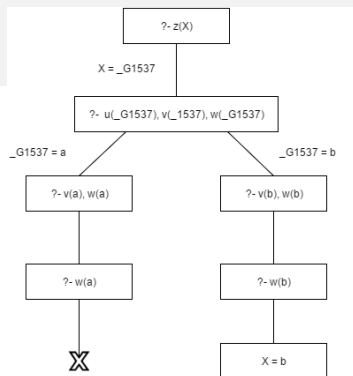
```
loves(henk, maria).  
loves(theo, maria).  
jealous(X, Y) :- loves(X, Z), loves(Y, Z).
```

Figure 2: Knowledge database



## A more complicated example (2)

```
[trace] 6 ?- z(X).  
Call: (7) z(_G1537) ? creep  
Call: (8) u(_G1537) ? creep  
Exit: (8) u(a) ? creep  
Call: (8) v(a) ? creep  
Exit: (8) v(a) ? creep  
Call: (8) w(a) ? creep  
Fail: (8) w(a) ? creep  
Redo: (8) u(_G1537) ? creep  
Exit: (8) u(b) ? creep  
Call: (8) v(b) ? creep  
Exit: (8) v(b) ? creep  
Call: (8) w(b) ? creep  
Exit: (8) w(b) ? creep  
Exit: (7) z(b) ? creep  
X = b.
```



## A more complicated example (3)

- ▶ Results are not always as expected
- ▶ `jealous(X,Y)`:

```
3 ?- jealous(X,Y)
X = Y, Y = henk ;
X = henk,
Y = theo ;
X = theo,
Y = henk ;
X = Y, Y = theo.
```

Figure 3: `jealous(X,Y)`

- ▶ `jealous(X,X)`

```
1 ?- jealous(X,X)
|
X = henk ;
X = theo.
```

Figure 4: `jealous(X,X)`





# Powerful basis for logical inference

- ▶ Combining unification and backtracking to search trees results in a fast tool for logical inference
- ▶ Understanding of underlying concepts is important to understand results produced
- ▶ Various implementations might grant different results, when considering a query like:

?- father(X) = X

