Praktikum 1

# SLD-Resolution

## Successor.pl ?- zahl(X).

### Regel:

### zahl(0).

### zahl(succ(X)) :- zahl(X).

ⱯX: zahl(0) ≡ zahl(0)

ⱯX: zahl(X) → zahl(succ(X)) ≡ ⱯX: ꓶzahl(X) ꓦ zahl(succ(X))) ≡ ꓶzahl(X) ꓦ zahl(succ(X)))

Klauselmenge R:= {zahl(0), ꓶzahl(X) ꓦ zahl(succ(X))}

### Anfrage ?- zahl(X). („Was sind alle Nachfolger von X“)

ꓱX: zahl(X) ≡ ꓱX: zahl(X)

### Zu beweisen:

Regel Ⱶ Anfrage

### Beweis

1. **zahl(0)**
2. **ꓶzahl(X) , zahl(succ(X))**

Anfrage negiert der Klauselmenge hinzufügen

1. ꓶꓱX: zahl(X) ≡ ⱯX: ꓶzahl(X) ≡ **ꓶzahl(X)**

3 und 1 X= 0

1. **Leere Klausel**
2. **REDO**, weil wir weitere Lösungen finden wollen
3. **zahl(0)**
4. **ꓶzahl(X) , zahl(succ(X))**
5. **ꓶzahl(X)**

3 und 2 X = succ(X)

1. **ꓶzahl(succ(X)))**

6 und 1. X = succ(0)

[debug] ?- zahl(X).

\* Call: (8) zahl(\_8196) ? creep

\* Exit: (8) zahl(0) ? creep

X = 0 ;

\* Redo: (8) zahl(\_8196) ? creep

\* Call: (9) zahl(\_8406) ? creep

\* Exit: (9) zahl(0) ? creep

\* Exit: (8) zahl(succ(0)) ? creep

X = succ(0) ;

8. **Leere Klausel**

1. **REDO**
2. **zahl(0)**
3. **ꓶzahl(X) , zahl(succ(X))**
4. **ꓶzahl(X)**
5. **ꓶzahl(succ(X)))**

3 und 4 X = succ(succ(X))

1. **ꓶzahl(succ(succ(X)))**

8 und 1 X = succ(succ(0))

1. **Leere Klausel**
2. **REDO**
3. **zahl(0)**
4. **ꓶzahl(X) , zahl(succ(X))**
5. **ꓶzahl(X)**
6. **ꓶzahl(succ(X)))**
7. **ꓶzahl(succ(succ(X)))**

Usw.

## Add.pl ?- add(succ(succ(0)), succ(0), Z).

### Regel:

### add(0, Y, Y)

### add(succ(X), Y, succ(Z):- add(X, Y, Z)

add(0, Y, Y) ≡ ꓯY: add(0, Y, Y) ≡ add(0,Y, Y)

add(succ(X), Y, succ(Z)):- add(X, Y, Z)

≡ ꓯX,Y, Z: add(X, Y, Z)→ add(succ(X), Y, succ(Z))

≡ꓯX, Y, Z: ꓶadd(X, Y, Z)ꓦadd(succ(X), Y, succ(Z))

≡ ꓶadd(X, Y, Z)ꓦ add(succ(X), Y, succ(Z))

Klauselmenge R:= { add(0,Y, Y), ꓶadd(X, Y, Z)ꓦ add(succ(X), Y, succ(Z))}

### Anfrage ?- add(succ(succ(0)), succ(0), Z). (“addiere succ(succ(0), also 2, mit succ(0), also 1 und schreibe das Ergebnis in Z”)

≡ ꓯZ: add(succ(succ(0)), succ(0), Z) → **false**

≡ ꓯZ:ꓶadd(succ(succ(0)), succ(0), Z)

\* Redo: (9) zahl(\_8406) ? creep

\* Call: (10) zahl(\_8410) ? creep

\* Exit: (10) zahl(0) ? creep

\* Exit: (9) zahl(succ(0)) ? creep

\* Exit: (8) zahl(succ(succ(0))) ? creep

X = succ(succ(0))

≡ ꓶadd(succ(succ(0)), succ(0), Z)

### Beweis

1. **add(0,Y, Y)**
2. **ꓶadd(X, Y, Z)ꓦ add(succ(X), Y, succ(Z))**

Anfrage negiert der Klauselmenge hinzufügen

1. **add(succ(succ(0)), succ(0), Z)**

3 und 2 X= succ(0), Y = succ(0), Z = succ(Z)

1. ꓶadd(succ(0), succ(0), succ(Z))

4 und 2 X= 0, Y= succ(0), Z= succ(succ(Z))

1. ꓶ(ꓶadd(0, succ(0), succ(succ(Z)

≡ add(0, succ(0), succ(succ(Z)

5 und 1 Y= succ(0), Z= succ(succ(succ(0))

1. **Leere Klausel**

[debug] ?- add(succ(succ(0)),succ(0),Z).

\* Call: (8) add(succ(succ(0)), succ(0), \_8504) ?

\* Call: (9) add(succ(0), succ(0), \_8746) ? creep

\* Call: (10) add(0, succ(0), \_8750) ? creep

\* Exit: (10) add(0, succ(0), succ(0)) ? creep

\* Exit: (9) add(succ(0), succ(0), succ(succ(0))) ? creep

\* Exit: (8) add(succ(succ(0)), succ(0), succ(succ(succ(0)))) ? creep

Z = succ(succ(succ(0))).