Praktikum 1

SLD-Resolution

Successor.pl ?- zahl(X).

[debug] ?- zahl(X).

Regel:

zahl(0).

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

∀X: zahl(0) ≡ zahl(0)

∀X: zahl(X) → zahl(succ(X)) ≡ ∀X: ˥zahl(X) V zahl(succ(X))) ≡ ˥zahl(X) V zahl(succ(X)))

Klauselmenge R:= {zahl(0), ˥zahl(X) V zahl(succ(X))}

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

∃X: zahl(X) ≡ ∃X: zahl(X)

Zu beweisen:

Regel ⊢ Anfrage

Beweis

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

   Anfrage negiert der Klauselmenge hinzufügen

3. ˥∃X: zahl(X) ≡ ∀X: ˥zahl(X) ≡ **˥zahl(X)**

3 und 1 X= 0

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

3 und 2 X = succ(X)

6. **˥zahl(succ(X)))**

6 und 1. X = succ(0)

* 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**
   7. **REDO**
      1. **zahl(0)**
      2. **¬zahl(X) , zahl(succ(X))**
      3. **¬zahl(X)**
      4. **¬zahl(succ(X)))**

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

   8. **¬zahl(succ(succ(X)))**

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

   9. **Leere Klausel**
   10. **REDO**
      1. **zahl(0)**
      2. **¬zahl(X) , zahl(succ(X))**
      3. **¬zahl(X)**
      4. **¬zahl(succ(X)))**
      5. **¬zahl(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) ≡ ∀Y: add(0, Y, Y) ≡ add(0,Y, Y)

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

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

≡∀X, Y, Z: ¬add(X, Y, Z)Vadd(succ(X), Y, succ(Z))

≡ ¬add(X, Y, Z)V add(succ(X), Y, succ(Z))

Klauselmenge R:= { add(0,Y, Y), ¬add(X, Y, Z)V
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”)

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

≡ ∀Z: ¬add(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))

≡ ˥add(succ(succ(0)), succ(0), Z)

Beweis

1. **add(0,Y, Y)**
2. **˥add(X, Y, Z)V add(succ(X), Y, succ(Z))**

   Anfrage negiert der Klauselmenge hinzufügen

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

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

4. ˥add(succ(0), succ(0), succ(Z))

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

5. ˥(˥add(0, succ(0), succ(succ(Z)
   ≡ add(0, succ(0), succ(succ(Z)

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

6. **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))).