Rekursion

- rekursive Prädikate
- deklarative vs. prozedurale Bedeutung von Prädikaten

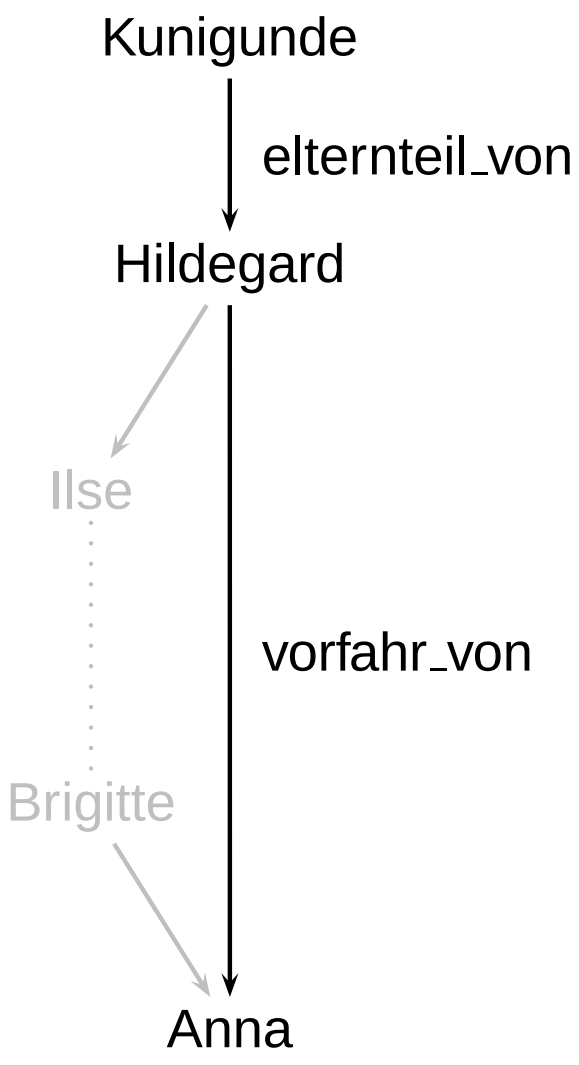
is_digesting/2

```
is_digesting(X,Y) :- just_ate(X,Y).
is_digesting(X,Y) :-
            just_ate(X,Z),
            is_digesting(Z,Y).

just_ate(mosquito,blood(john)).
just_ate(frog,mosquito).
just_ate(stork,frog).
```

Was antwortet Prolog auf die Anfragen:

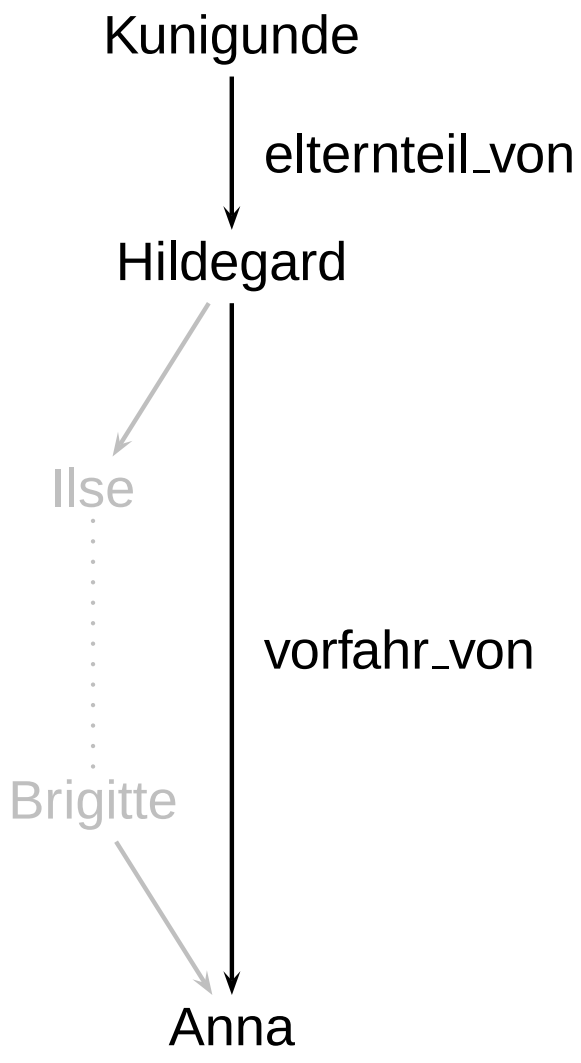
```
?- is_digesting(stork,frog).

?- is_digesting(stork,mosquito).

?- is_digesting(frog,X).
```

Rekursion: Blickwinkel 1



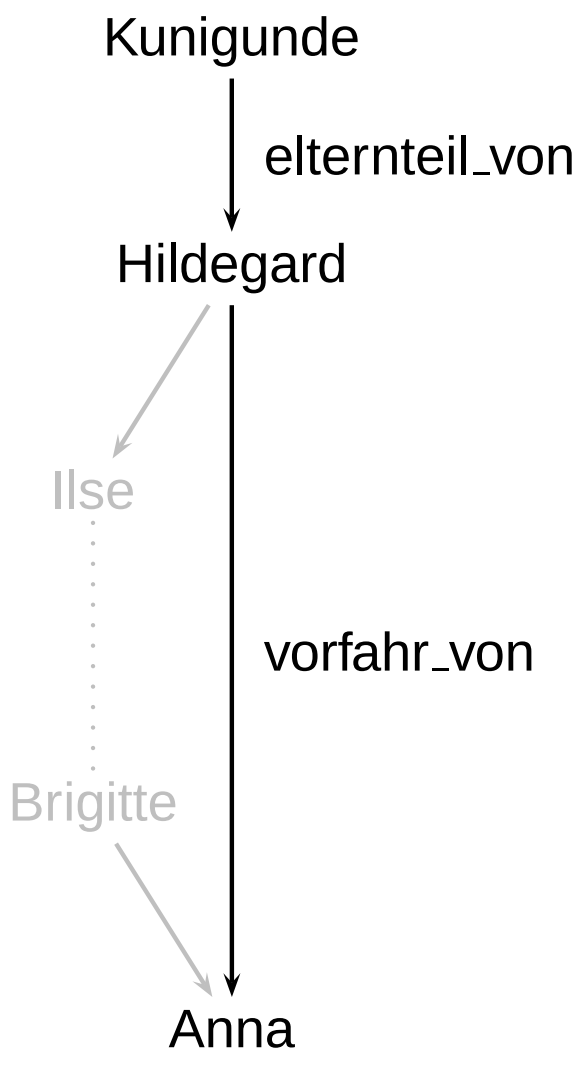
Ist Kunigunde ein Vorfahr von Anna?

Rekursion: Blickwinkel 1



Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde die Mutter von Hildegard ist, die ein Vorfahr von Anna ist.

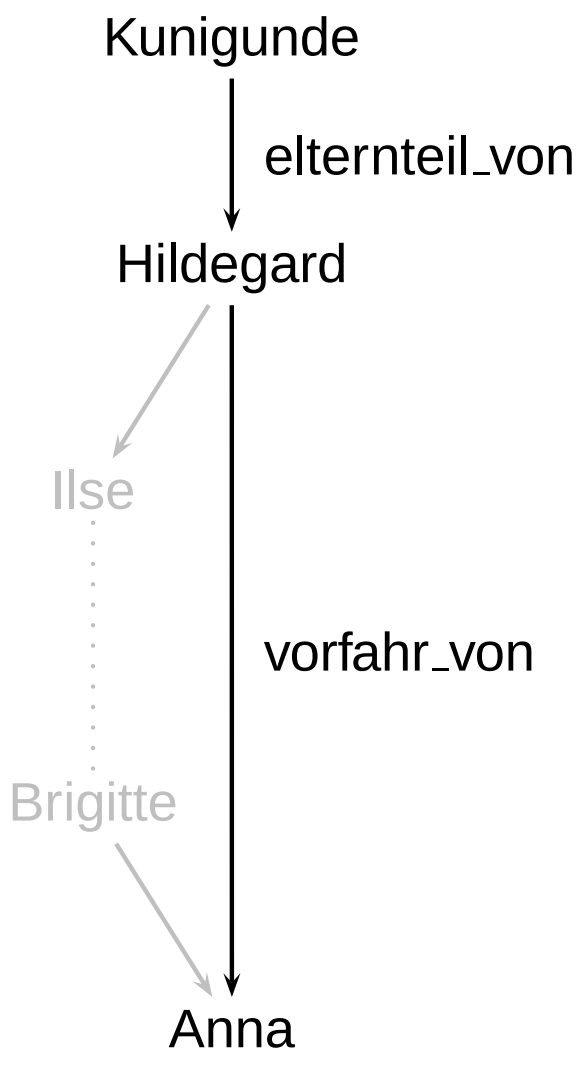
Rekursion: Blickwinkel 1



Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde die Mutter von Hildegard ist, die ein Vorfahr von Anna ist.

```
vorfahr_von(K,A) :-
      elternteil_von(K,H),
      vorfahr_von(H,A).
```

Rekursion: Blickwinkel 1

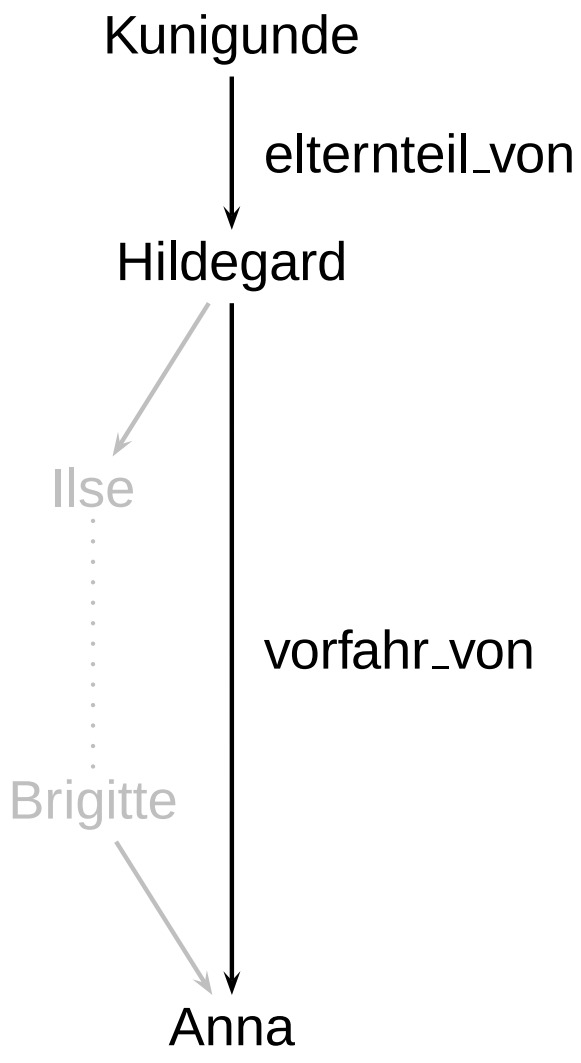


Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde die Mutter von Hildegard ist, die ein Vorfahr von Anna ist.

```
vorfahr_von(K,A) :-
        elternteil_von(K,H),
        vorfahr_von(H,A).
```

Reicht das?

Rekursion: Blickwinkel 1

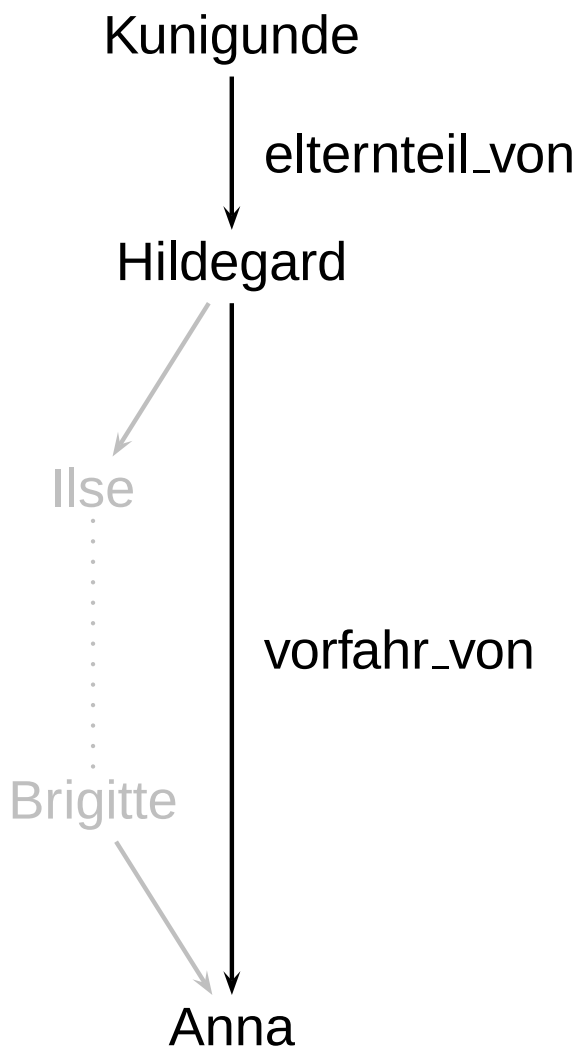


Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde die Mutter von Hildegard ist, die ein Vorfahr von Anna ist.

```
vorfahr_von(K,A) :-
    elternteil_von(K,H),
    vorfahr_von(H,A).
```

Reicht das? – Nein, weil Kunigunde auch ein Vorfahr von Hildegard ist.

Rekursion: Blickwinkel 1



Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde die Mutter von Hildegard ist, die ein Vorfahr von Anna ist.

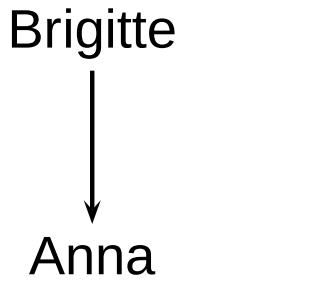
```
vorfahr_von(K,A) :-
    elternteil_von(K,H),
    vorfahr_von(H,A).
```

Reicht das? – Nein, weil Kunigunde auch ein Vorfahr von Hildegard ist.

```
vorfahr_von(K,H) :-
    elternteil_von(K,H).
```

Rekursion: Blickwinkel 2

Ist Brigitte ein Vorfahr von Anna?

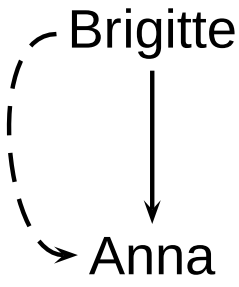


Elternteil: ⟶

Vorfahr: - - - →

Rekursion: Blickwinkel 2

Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.



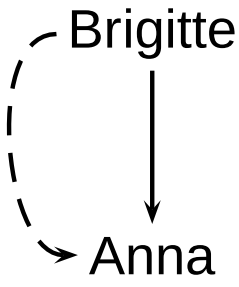
Elternteil: ⟶

Vorfahr: - - - →

Rekursion: Blickwinkel 2

Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

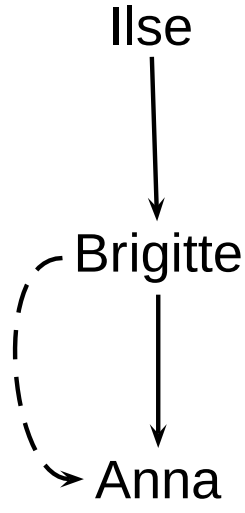
```
vorfahr_von(X,Y) :-
    elternteil_von(X,Y).
```



Elternteil: ⟶

Vorfahr: - - - →

Rekursion: Blickwinkel 2



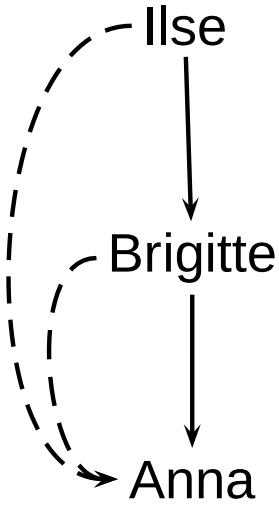
Elternteil: ⟶

Vorfahr: - - - →

Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
     elternteil_von(X,Y).
```

Rekursion: Blickwinkel 2



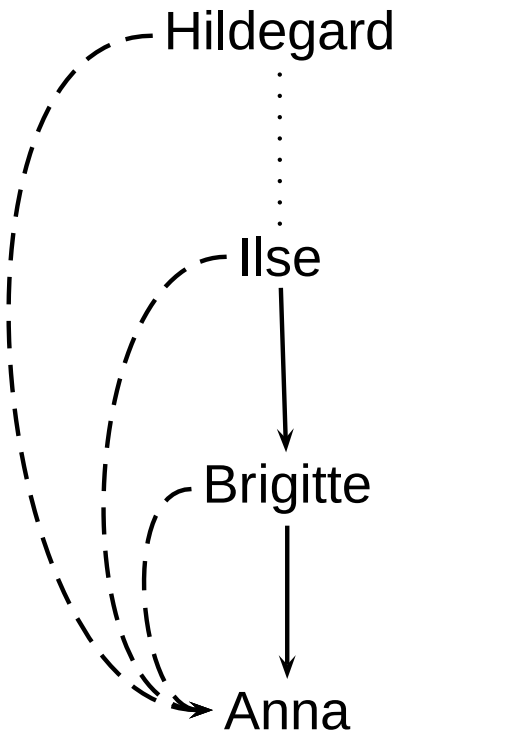
Elternteil: ⟶

Vorfahr: - - - →

Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
     elternteil_von(X,Y).
```

Rekursion: Blickwinkel 2



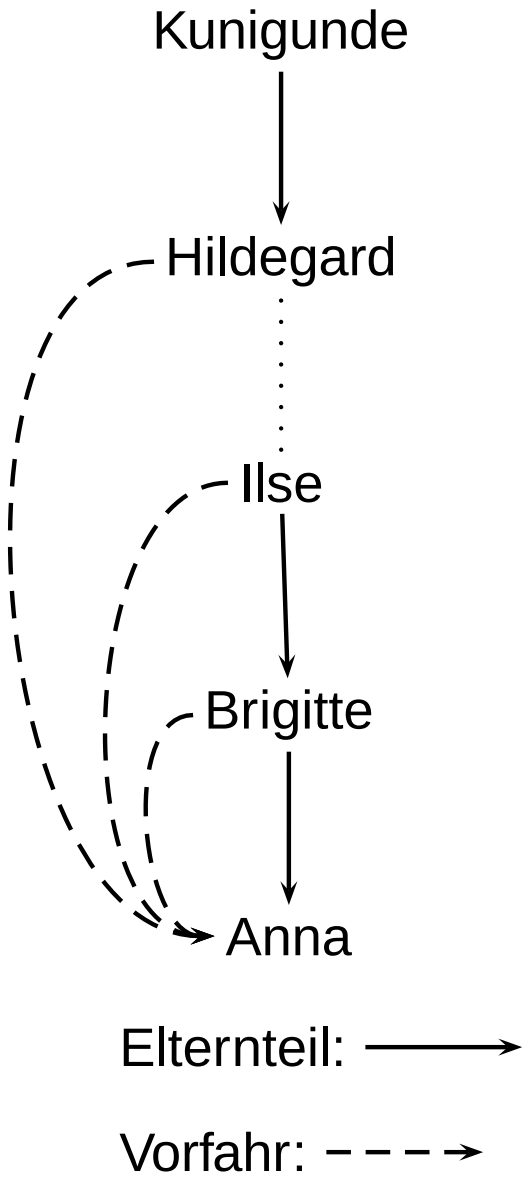
Elternteil: ⟶

Vorfahr: - - - →

Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
    elternteil_von(X,Y).
```

Rekursion: Blickwinkel 2

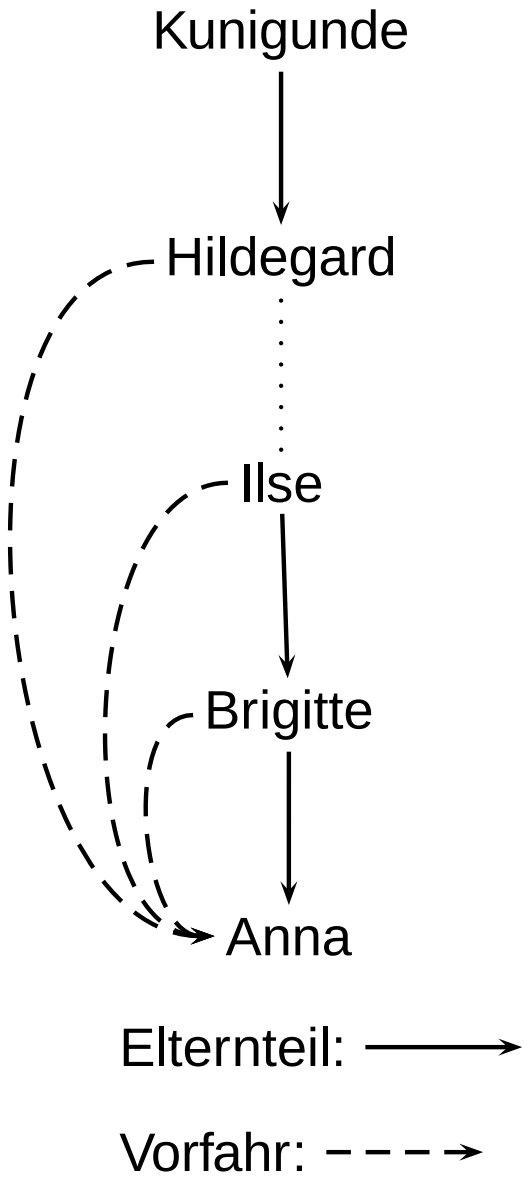


Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
    elternteil_von(X,Y).
```

Ist Kunigunde ein Vorfahr von Anna?

Rekursion: Blickwinkel 2

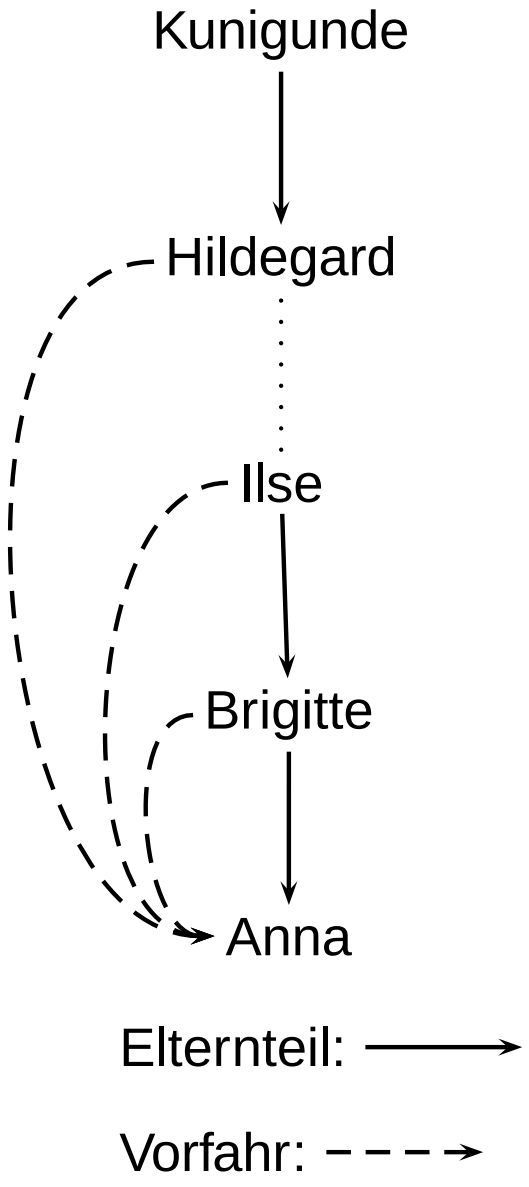


Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
    elternteil_von(X,Y).
```

Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde ein Elternteil von Hildegard ist, die ein Vorfahr von Anna ist.

Rekursion: Blickwinkel 2



Ist Brigitte ein Vorfahr von Anna? – Ja, weil Brigitte ein Elternteil von Anna ist und Elternteile sind Vorfahren.

```
vorfahr_von(X,Y) :-
    elternteil_von(X,Y).
```

Ist Kunigunde ein Vorfahr von Anna? – Ja, weil Kunigunde ein Elternteil von Hildegard ist, die ein Vorfahr von Anna ist.

```
vorfahr_von(K,A) :-
    elternteil_von(K,H),
    vorfahr_von(H,A).
```

Verarbeitung in Prolog

```
elternteil_von(kunigunde,hildegard).
elternteil_von(hildegard,ilse).
elternteil_von(ilse,brigitte).
elternteil_von(brigitte,anna).

vorfahr_von(X,Y) :- elternteil_von(X,Y).
vorfahr_von(X,Y) :-
        elternteil_von(X,Z),
        vorfahr_von(Z,Y).
```

```
?- vorfahr_von(hildegard,ilse).
?- vorfahr_von(kunigunde,anna).
?- vorfahr_von(hildegard,X).
```

Beispiel: Eine Form von Zählen

$0 \Rightarrow$ `0`

$1 \Rightarrow$ `succ(0)`

$2 \Rightarrow$ `succ(succ(0))`

$3 \Rightarrow$ `succ(succ(succ(0)))`

$\vdots$

Ziel: Ein Prädikat `number/1`, das testet, ob das Argument eine Zahl in der `succ`-Darstellung ist.

Beispiel: Eine Form von Zählen

$0 \Rightarrow$ `0`

$1 \Rightarrow$ `succ(0)`

$2 \Rightarrow$ `succ(succ(0))`

$3 \Rightarrow$ `succ(succ(succ(0)))`

$\vdots$

Ziel: Ein Prädikat `number/1`, das testet, ob das Argument eine Zahl in der `succ`-Darstellung ist.

`0` ist eine Zahl.

Beispiel: Eine Form von Zählen

$0 \Rightarrow$ `0`

$1 \Rightarrow$ `succ(0)`

$2 \Rightarrow$ `succ(succ(0))`

$3 \Rightarrow$ `succ(succ(succ(0)))`

$\vdots$

Ziel: Ein Prädikat `number/1`, das testet, ob das Argument eine Zahl in der `succ`-Darstellung ist.

`0` ist eine Zahl.

Wenn X eine Zahl ist, dann

ist `succ(X)` auch eine Zahl.

Beispiel: Eine Form von Zählen

$0 \Rightarrow$ `0`

$1 \Rightarrow$ `succ(0)`

$2 \Rightarrow$ `succ(succ(0))`

$3 \Rightarrow$ `succ(succ(succ(0)))`

$\vdots$

Ziel: Ein Prädikat `number/1`, das testet, ob das Argument eine Zahl in der `succ`-Darstellung ist.

`0` ist eine Zahl. `number(0).`

Wenn X eine Zahl ist, dann ist `succ(X)` auch eine Zahl.

Beispiel: Eine Form von Zählen

$0 \Rightarrow$ `0`

$1 \Rightarrow$ `succ(0)`

$2 \Rightarrow$ `succ(succ(0))`

$3 \Rightarrow$ `succ(succ(succ(0)))`

$\vdots$

Ziel: Ein Prädikat `number/1`, das testet, ob das Argument eine Zahl in der `succ`-Darstellung ist.

`0` ist eine Zahl.	`number(0).`
Wenn X eine Zahl ist, dann ist `succ(X)` auch eine Zahl.	`number(succ(X)) :- number(X).`

Beispiel: Addieren

Ziel: Ein Prädikat `add/3`, das drei Zahlen in `succ`-Darstellung als Argumente nimmt. Das dritte Argument soll die Summe der beiden ersten sein. Z.B.

```
?- add(succ(0), succ(succ(0)), X).
X = succ(succ(succ(0)))
yes
?- add(succ(succ(0)), succ(0), X).
X = succ(succ(succ(0)))
yes
?- add(0, succ(succ(0)), X).
X = succ(succ(0))
yes
```

Beispiel: Addieren (2)

Basisfall:

Wenn das erste Argument `0` ist, dann ist das Ergebnis gleich dem dritten Argument.

Rekursiver Fall:

Wenn die Summe von `X` und `Y` gleich `Z` ist, dann ist die Summe von `succ(X)` und `Y` gleich `succ(Z)`.

Aufgaben

1. Definiert das Prädikat `add/3` in Prolog.
2. Definiert ein Prädikat `greater_than/2`, das als Argumente zwei Zahlen in `succ`-Darstellung nimmt und testet, ob die erste größer ist als die zweite.
3. Wir haben die folgende Wissensbasis:

   ```
   directTrain(forbach,saarbruecken).
   directTrain(freyming,forbach).
   directTrain(fahlquemont,stAvold).
   directTrain(stAvold,forbach).
   directTrain(metz,fahlquemont).
   ```

   Schreibt ein Prädikat `travel/2`, das definiert zwischen welchen zwei Städten es eine Zugverbindung gibt, bei der direkte Zugverbindungen aneinandergehängt werden.

Deklarative vs. Prozedurale Bedeutung (1)

```
number(0).
number(succ(X)) :- number(X).
```

```
number(succ(X)) :- number(X).
number(0).
```

Deklarative vs. Prozedurale Bedeutung (1)

```
number(0).
number(succ(X)) :- number(X).
```

```
number(succ(X)) :- number(X).
number(0).
```

Die beiden Versionen von `number/1` haben

- die gleiche deklarative Bedeutung
- und verschiedene prozedurale Bedeutungen.

```
?- number(X).
```

Deklarative vs. Prozedurale Bedeutung (2)

```
vorfahr_von(X,Y) :- elternteil_von(X,Y).
vorfahr_von(X,Y) :- elternteil_von(X,Z),
                    vorfahr_von(Z,Y).
```

```
vorfahr_von(X,Y) :- elternteil_von(X,Y).
vorfahr_von(X,Y) :- vorfahr_von(Z,Y),
                    elternteil_von(Z,Y).
```

```
?- vorfahr_von(anna, kunigunde).
```

```
elternteil_von(kunigunde,hildegard).
elternteil_von(hildegard,ilse).
elternteil_von(ilse,brigitte).
elternteil_von(brigitte,anna).
```

Deklarative vs. prozedurale Bedeutung

deklarative Bedeutung:

die logische Bedeutung einer Prolog-Wissensbasis

prozedurale Bedeutung:

wie eine Wissensbasis von Prolog verarbeitet wird

Noch ein Beispiel

```
p :- p
```

Noch ein Beispiel

```
p :- p
```

deklarativ: Wenn p dann p.

prozedural: Bei der Anfrage p gerät Prolog in eine Endlosschleife.

Zusammenfassung

- Rekursion ist eine äußerst wichtige Programmiertechnik (nicht nur in Prolog).
- Mit Hilfe von Rekursion (plus Matching) können kompakte und elegante Programme geschrieben werden.
- Es ist wichtig, sich auch die prozedurale Bedeutung von Programmen klarzumachen.

Wichtige Begriffe: Basisklausel, rekursive Klausel, deklarative und prozedurale Bedeutung

Nächste Woche: Listen: eine rekursive Struktur

Übungsaufgaben: Das Übungsblatt ist auf der Web-Seite.