Einführung in Prolog

- Prolog (vom Französischen: **Pro**grammation en **Log**ique, deutsch: **Pro**grammieren in **Log**ik)
- Prolog-Programme bestehen aus einer Datenbasis, deren Einträge **Fakten** und **Regeln** genannt werden.
- Der Nutzer formuliert **Anfragen**, auf die der Interpreter systematisch durch Nutzung dieser Fakten und Regeln eine Antwort findet.
- Ein positives Ergebnis bedeutet, dass die Anfrage ableitbar war.

Einführung in Prolog: Fakten

- Albert ist männlich:
  `male(albert).`
- Berti ist ein Elternteil von Albert:
  `parent(berti,albert).`

- Ein Fakt besteht aus einem Prädikat und dessen Argumenten.
- Statements werden immer mit Punkt abgeschlossen.
- Variablen beginnen mit Großbuchstaben.

- Albert ist Kind aller:
  `parent(X,albert).`

Einführung in Prolog: Regeln

- Alles, was nicht männlich ist, ist weiblich:
  `female(X) :- not(male(X)).`

- Eine Regel beschreibt die Abhängigkeit eines Fakts von einem oder mehreren anderen Fakten.
- Der Regeloperator `:-` ist dabei wie ein umgedrehter Implikationspfeil zu lesen.
- Das Komma kann dabei als Und-Operator verwendet werden.
- Haben zwei Regeln die gleiche Konsequenz, folgt diese, wenn mindestens in einer Regel die Bedingung erfüllt ist.

- Wenn X Elternteil von Y und männlich ist, dann ist X Vater von Y: `father(X,Y) :- parent(X,Y), male(X).`

Einführung in Prolog: Anfragen

- Ist Albert männlich?
  `?- male(albert).`
- Eltern von Albert?
  `?- parent(X,albert).`
- Alle Eltern mit Kindern?
  `?- parent(X,Y).`

- Anfragen werden im Prolog-Interpreter gestellt.
- In Anfragen kann ebenso mit Variablen gearbeitet werden, um alle erfüllenden Belegungen zu finden.

Übung 2

- Worin unterscheiden sich `ancestor(X,Y)` und `ancestor2(X,Y)`?

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(Z,Y), ancestor(X,Z).

ancestor2(X,Y) :- parent(X,Y).
ancestor2(X,Y) :- ancestor2(X,Z), parent(Z,Y).
```

- Spielt die Reihenfolge der Klauseln eine Rolle?
  - Aus logischer Sicht: Nein.
  - Für die Abarbeitung in Prolog: Ja. Führt man z.B. die Anfrage `?- ancestor2(berti,berti).` aus, erhält man den Fehler „Out of local stack“. Die Ursache liegt darin, dass `ancestor2(X,Z).` permanent aufgerufen wird, was zum Überlauf führt.
- Regeln sollten also so aufgebaut werden, dass einfache Berechnungen zu Beginn stehen und rekursive Aufrufe zuletzt erfolgen, um Nicht-Terminierung und Speicherprobleme zu vermeiden.

Zusatzaufgabe 1 (a)

$$(\lambda z\, \underbrace{x.zx(\lambda y.yx)}_{GV=\{x,y\}})\ \underbrace{(\lambda y.zx)}_{FV=\{x,z\}}\ (\lambda z.z)$$

$$\Rightarrow_\alpha\ (\lambda z\, \underbrace{x_1.zx_1(\lambda y.yx_1)}_{GV=\{x_1,y\}})\ \underbrace{(\lambda y.zx)}_{FV=\{x,z\}}\ (\lambda z.z)$$

$$\Rightarrow_\beta\ (\lambda x_1.(\lambda y.\ \underbrace{zx}_{GV=\emptyset}\ ))\ \underbrace{x_1}_{FV=\{x_1\}}\ (\lambda y.yx_1))(\lambda z.z)$$

$$\Rightarrow_\beta\ (\lambda x_1.\ \underbrace{zx(\lambda y.yx_1)}_{GV=\{y\}})\ \underbrace{(\lambda z.z)}_{FV=\emptyset}$$

$$\Rightarrow_\beta\ zx(\lambda y.y(\lambda z.z))$$

Zusatzaufgabe 1 (b)

$$\langle F \rangle = \Big(\lambda f\, x\, y.\langle \mathsf{ite} \rangle \Big(\langle \mathsf{iszero} \rangle \Big(\langle \mathsf{pred} \rangle\, x \Big) \Big) \Big(\langle \mathsf{mult} \rangle\, \langle 2 \rangle\, y \Big) \Big(\langle \mathsf{add} \rangle\, y \Big(\langle \mathsf{mult} \rangle \big(\langle \mathsf{succ} \rangle\, x \big) \big(f\, \big(\langle \mathsf{pred} \rangle\, x \big)\, \big(\langle \mathsf{add} \rangle x\, y \big) \big) \Big) \Big) \Big)$$

$$\langle f \rangle = \langle Y \rangle \langle F \rangle$$

Zusatzaufgabe 1 (c)

$$
\begin{aligned}
& \langle Y\rangle\langle G\rangle = (\lambda z.((\lambda u.z(uu))(\lambda u.z(uu))))\langle G\rangle \\
\Rightarrow_\beta\ & ((\lambda u.\langle G\rangle(uu))(\lambda u.\langle G\rangle(uu))) = \langle Y_G\rangle \\
\Rightarrow_\beta\ & \langle G\rangle((\lambda u.\langle G\rangle(uu))(\lambda u.\langle G\rangle(uu))) = \langle G\rangle\langle Y_G\rangle
\end{aligned}
$$

$$
\begin{aligned}
& \langle Y\rangle\langle G\rangle\langle 3\rangle\langle 0\rangle \\
\Rightarrow^*\ & \langle G\rangle\langle Y_G\rangle\langle 3\rangle\langle 0\rangle \\
\Rightarrow^*\ & \langle \mathsf{ite}\rangle \underbrace{(\langle \mathsf{iszero}\rangle\langle 0\rangle)}_{\Rightarrow^*\langle \mathsf{true}\rangle} \underbrace{(\langle \mathsf{succ}\rangle\langle 3\rangle)}_{\Rightarrow^*\langle 4\rangle}(\ \dots\ ) \\
\Rightarrow^*\ & \langle 4\rangle
\end{aligned}
$$