First-Argument-Indexing

Effizienteres Beweisen

```
% lex_v(Lemma, Hilfsverb, Subcat)
lex_v(aalen, haben, [akk]).
%... 6000 weitere Klauseln
lex(züngeln, haben, []).
```

- Der Interpreter muss bei jedem Beweisschritt feststellen, welcher Klauselkopf mit einem Ziel unifiziert.
- Naives Vorgehen
  - Interpreter probiert der Reihe nach alle Klauseln eines Prädikats durch. Je mehr Klauseln ein Prädikat hat, umso aufwändiger wird das Durchprobieren.
- Effizienteres Vorgehen mit First-Argument-Indexing
  - Hauptfunktor des 1. Arguments des Prädikats wird als Index verwendet, mit dem die passenden Klauseln direkt angesprungen werden können.
  - Das Abspeichern der Klauseln wird etwas aufwändiger, dafür ist der Aufwand beim Suchen eines Klauselkopfs nicht mehr abhängig von der Anzahl Klauseln eines Prädikats!

Die Bibliotheksmetapher

Die Bibliothek als Wissensbasis

- Eine Bibliothek besteht aus Büchern, ein Programm besteht aus Prädikaten.
- Ein Buch enthält Textabschnitte, ein Prädikat enthält Klauseln.

Indexgestütztes Nachschlagen

- In Büchern mit einem Index (Stichwortregister) kann ich gezielt Textabschnitte nachschlagen ohne das ganze Buch zu lesen. In Programmen mit First-Argument-indizierten Klauseln kann der Interpreter Klauseln nachschlagen ohne die ganze Prädikatsdefinition zu durchsuchen.

Adressierung

- Stichwortregister adressieren typischerweise Seitenzahlen oder Kapitel. First-Argument-Indizes adressieren die entsprechenden Speicherstellen durch Anwendung einer Hash-Funktion auf den Hauptfunktor.

Instantiierung und Reihenfolge

1. Argument als Inputargument

- Damit aus dem Hauptfunktor die Speicherstelle berechnet werden kann, muss er beim Aufruf eines Ziels instantiiert sein!
- **Vermeide** Output-Argumente an erster Argumentsposition: Inputargumente sollten immer vor Output-Argumenten erscheinen.
  - Bei Prädikaten ohne eindeutige Input-/Output-Zuordnung an die Argumente müssen allenfalls 2 Hilfsprädikate definiert werden, die first-argument-optimiert sind.

1. Argument als Fallunterscheidung

- Die Fallunterscheidung von Klauseln sollte möglichst durch Hauptfunktoren im Klauselkopf ausgedrückt werden.
- **Vermeide** catch-all-Klauseln: Klauseln, deren 1. Argument eine Variable ist, sollten möglichst vermieden werden.

Beispiel: Effizienzgewinn

Standardreihenfolge von Argumenten

- Ein 1. Argument, das mit instantiiertem und fallunterscheidendem Hauptfunktor aufgerufen wird, optimiert ein Programm messbar.

```
reverse(List, Rev) :-
   reverse(List, [], Rev).

reverse([], Akku, Akku).
reverse([X|Rest], Akku, Rev) :-
   reverse(Rest, [X|Akku], Rev).
```

```
?- length(_L, 20000),
   time(reverse(_L, _LR)).
Elapsed time: 67 ms
```

```
reverse2(List, Rev) :-
   reverse2(Rev, List, []).

reverse2(Akku,[], Akku).
reverse2(Rev, [X|Rest], Akku) :-
   reverse2(Rev, Rest, [X|Akku]).
```

```
?- length(_L, 20000),
   time(reverse2(_L, _LR)).
Elapsed time: 100 ms
```