Last-Call-Optimization

Effizienteres Beweisen

- Der Interpreter muss bei jedem Aufruf eines Prädikats ein sog. *Stack Frame* im Speicher erstellen. Über dieses *Stack Frame* werden die Instantiierungen der Variablen des Prädikataufrufs verwaltet.

```
list_length([], 0).
list_length([_|Rest], N) :-
    list_length(Rest, M),
    N is M + 1.
```

- Der verschachtelte Aufruf von rekursiven Prädikaten führt schnell zu einer Vielzahl von *Stack Frames*.
  - Für jedes Element in der Liste wird ein Stack Frame angelegt bei list_length/2!

Naives Vorgehen

- Das *Stack Frame* bleibt bestehen, bis die allerletzte Lösung des Aufrufs berechnet wurde.

Effizienteres Vorgehen

Last-Call-Optimization

- Das *Stack Frame* mit den lokal benötigten Datenstrukturen eines Prädikats *P*

$$P :- R_1,\ldots, R_n$$

kann **vor** dem letzten Unteraufruf R_n (*last call*) freigegeben werden, falls gilt:

- Es gibt keine weitere Klausel für *P*.
- Es gibt keine Entscheidungspunkte für die Unteraufrufe $R_1,\ldots,R_{n-1}$.

Last-Call-Optimization ermöglichen

Um Last-Call-Optimierung nicht bloss in der letzten Klausel zu ermöglichen:

- Nütze **First-Argument-Indexing** aus: Prolog erkennt so, dass auch andere als die letzte Klausel Last-Call-optimierbar sind.
- Ein **Cut** vor dem letzten Aufruf in einer Klausel ermöglicht immer Last-Call-Optimierung.

$$P :- R_1,\ldots, !, R_n$$

  - Der Cut kostet allerdings auch wieder etwas Zeit!
- Mache Prädikate möglichst **endrekursiv**! D.H. der rekursive Aufruf kommt in der letzten Klausel als letzter Aufruf.
  - Um Rechtsrekursion (oder Endrekursion) zu erhalten, muss manchmal ein Akkumulator eingeführt werden, der Zwischenresultate speichert!

Listenlänge mit Endrekursion

Endrekursion dank Akkumulator

- Damit Last-Call-Optimierung auf den rekursiven Aufruf möglich wird, speichert ein zusätzliches Argument die Anzahl bisher angetroffener Elemente.

```
list_length_accu(List, Length) :-
    list_length_accu(List, 0, Length).

list_length_accu([], Accu, Accu).
list_length_accu([_|Rest], Accu, N) :-
    NewAccu is  Accu + 1,
    list_length_accu(Rest, NewAccu, N).
```

- Dieses Prädikat braucht weniger Speicher, da dasselbe *Stack Frame* für alle rekursiven Aufrufe verwendbar ist.