Einführung in das Programmieren – Prolog
Sommersemester 2006

Teil 6: Verschiedenes

Version 1.0

Gliederung der LV

Teil 1: Ein motivierendes Beispiel

Teil 2: Einführung und Grundkonzepte
- Syntax, Regeln, Unifikation, Abarbeitung

Teil 3: Arithmetik

Teil 4: Rekursion und Listen

Teil 5: Programmfluß
- Negation, Cut

Teil 6: Verschiedenes
- Ein-/Ausgabe, Programmierstil

Teil 7: Wissensbasis
- Löschen und Hinzufügen von Klauseln

Teil 8: Fortgeschrittene Techniken
- Metainterpreter, iterative Deepening, PTTP, Differenzlisten, doppelt verkettete Listen

Programmierstil

Layout

- Kommentare
  1. Alle Programme sollten ausführlich kommentiert sein.
  2. Kommentare helfen stark zum Verständnis und dienen zur Wartung des Programms
- Aufteilung des Programms in mehrere Dateien; in diesem Fall Information über Nutzen in header mitaufnehmen.
- Kommentiere auch einzelne Prädikatsdefinitionen

```
append([], L, L]).                  % base case
append([X|L1], L2, [X|L3]) :-
      append(L1, L2, L3).           % recursion on first argument
```

Programmierstil

- Aufbau des Codes
  - jedes Prädikat in eine eigene Zeile
  - auch Klammern, Semikolon, $->$ in eigene Zeile
  - Semikolon und $->$ weniger einrücken, um sie hervorzuheben
  - setze Klammern, sobald Zweifel bestehen könnte

Programmierstil

- Vermeide Seiteneffekte wo immer möglich
- Programm sollte möglichst nicht zur Laufzeit verändert werden
- Die Verwendung von Cuts sollte mit großer Sorgfalt erfolgen
- `;` (Oder Operator): Sollte möglichst vermieden werden.

  ```
  a :- b;c.
  ```

  sollte ersetzt werden mit

  ```
  a :- b.
  a :- c.
  ```

Ein- und Ausgabe

- Verschiedene Standardprädikate dienen der Ein- und Ausgabe von Objekten.

Man hat Prädikate für unterschiedliche Ebenen von Objekten:

- Regeln
- Terme
- Zeichen/Bytes/...

Streams

Ein-/Ausgabe streambasiert

open, **close** Stream auf/von Datei öffnen/schließen

Einfachere Variante

see(file), **tell(file)** Alle Leseoperationen/Schreiboperationen beziehen sich jetzt auf das Lesen/Schreiben vom `file`

Einlesen

read_clause Lies eine Regel ein

read_term, **read** Lies einen Term ein

get_byte Liest ein Byte ein

get_code, **get_char**, Liest ein Zeichen ein

at_end_of_stream

Schreiben

portray_clause Schreibt eine Regel ”schön”

write Schreibt Term (Operatordefinitionen, z.B. infix, werden beachtet)

write_canonical Schreibt Term in kanonischer Form (d.h. $f(\cdots)$ für alle Funktoren)

format, **sformat** Formatierte Ausgabe auf Stream bzw. in String

```
format(+Formatstring, +Liste\_von\_Argumenten)
```

`Formatstring`: steuert die Ausgabe

- ∼a: entsprechendes Argument ist ein Atom
- ∼c: entsprechendes Argument ist integer, ausgedruckt wird zugehöriger ASCII-Wert
- ∼d: entsprechendes Argument ist Dezimalzahl
- ∼n: Newline
- . . . Setze Tabs, steuere Anzahl der Nachkommastellen, . . .

Beispiel:

```
?- format('1. Argument: ~a~nASCII von ~c ist ~d',
          [hallo, 66, 66]).
1. Argument: hallo
ASCII von B ist 66
```

Quellfiles

consult Lädt Prolog-File

Abkürzung: [file]

ensure_loaded Lädt File, aber nur beim ersten Aufruf

use_module Lädt Modul (library)