Programmierkurs Prolog

SS 1998

Thorsten Joachims

Universitaet Dortmund

LS VIII - Prof. K. Morik

Compiler/Debugger

Compiler

- Kommandos
- Compiler-Modes
- Mode-Deklarationen

Debugger

- Erweitertes Box-Modell
- Tracing
- Interaktive Kommandos

Compiler

Kompiliert in virtuellen Code, der von einer abstrakten Prolog-Maschine ausgeführt wird.

Direktiven:

- beginnen mit :-/1 oder ?-/1
- werden beim kompilieren ausgeführt

Klauseln

- der Rest
- werden kompiliert
- Prädikate mit mehreren Klauseln sollten immer zusammenhängend im File stehen
- Klauseln werden static kompiliert, wenn nicht explizit durch dynamic/1 anders gewünscht

Compiler Kommandos

- `compile(File)`
  Kompiliert den Inhalt des Files.
- `compile(File,Modul)`
  Kompiliert den Inhalt des Files in das Modul.
- `compile_stream(Stream)`
  Kompiliert aus Eingabestrom bis EOF
- `compile_term(Clauses)`
  Kompiliert Liste von Klauseln und Direktiven.
- `dump(File)`
  Schreibt Dump der Klauselbasis in File.
- `ensure_loaded(File)`
  Kompiliert File, falls die noch nicht in der neuesten Version kompiliert ist.
- `make`
  Rekompiliert alle zur Zeit geladenen Files, die zwischenzeitlich geändert wurden.
- `lib(File)`
  Lädt die angegebene Library (aus library-path)
- `assert(Clause)`

Compiler Modes

Modes werden durch Flags gesetzt. Flags setzt man mit

```
set_flag(+Flag,on/off).
```

Modes:

- `debug_compile`: Erzeugt Code zur Inspektion durch den Debugger (`on/off`).
- `occur_check`: Schaltet den Occur-Check ein/aus (`on/off`).
- `dfid_compile`: Kompiliert, so daß die dfid-Library zur Iterative-Deepening Suche benutzt werden kann (`on/off`).
- `float_precision`: Fließkomma-Genauigkeit (`single/double`).
- `all_dynamic`: Alle Prädikate werden dynamisch kompiliert (`on/off`).
- `variable_names`: Merkt sich die Namen der Variablen für das Debugging (`on/off`).

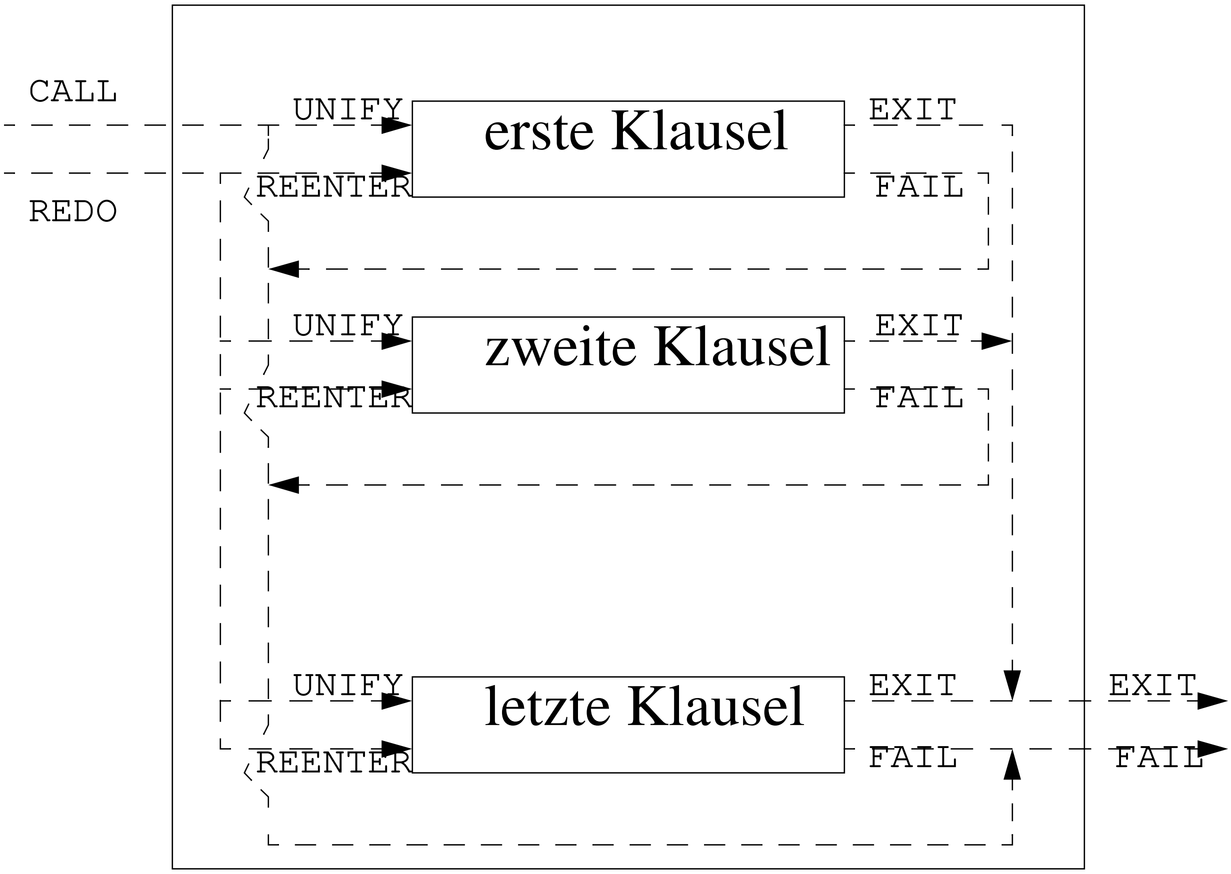
Mode-Deklarationen

Mode-Deklarationen geben dem Compiler zusätzliche Informationen zur Optimierung des Codes.

```
:- mode p(+), q(-), r(++,?).
```

+	Das Argument ist instantiiert (also keine Variable).
++	Das Argument ist grundinstantiiert.
–	Das Argument ist nicht instantiiert (also eine freie Variable, die in keinem anderen Argument auftaucht).
?	sonst

Boxen-Modell




```
S+(4)  2  *EXIT module:foo(one,X,two)  (dbg)?-
12  3   4  5  6      7          8                 9
```

Format der Trace-Zeilen

```
S+(4)  2  *EXIT module:foo(one,X,two)  (dbg)?-
12 3   4  5  6      7         8                 9
```

1. C: externe Prozedur
   S: skipped Prozedur
   N: Prozedur, die nicht in Debug-Modus komp. ist
2. Das + zeigt an, daß auf die Prozedur ein Spy-Point gesetzt ist.
3. Invocation-Number (fortlaufend pro Box)
4. Tiefe
5. Der * zeigt an, daß noch Choice-Points existieren.
6. Port-Name
7. Modul-Name
8. Goal
9. Debugger-Prompt

Aufruf des Debuggers

`trace/0`

Alle nachfolgenden Ziele werden im Creep-Modus ausgeführt

`debug/0`

Alle nachfolgenden Ziele werden in Leap-Modus ausgeführt

`spy/1`

Setze Spy-Point.

`nospy/1`

Entferne Spy-Point.

`notrace/0`

Schaltet trace aus.

`nodebug/0`

Schaltet debug aus.

Interaktive Kommandos

`c`	creep: Gehe zum nächsten Port (Einzelschritt).
`s`	skip: Gehe zum nächsten Exit Port dieser Box.
`l`	leap: Gehe zum nächsten Port einer Prozedur mit Spy-Point.
`i V`	variable skip: Gehe bis zur Instantiierung von V.
`+`	Setze Spy-Point auf aktuelles Ziel.
`-`	Entferne Spy-Point von aktuellem Ziel.
`A`	up: Gehe zum Vater des aktuellen Ziels.
`B`	down: Gehe zum Sohn des aktuellen Ziels.
`C`	right: Gehe zum nächsten Teilziel nach rechts.
`D`	left: Gehe zum nächsten Teilziel nach links.
`R`	retry: Führe das aktuelle Ziel erneut aus.
`n`	nodebug: Debugging ausschalten.