Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

1,

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

Langfristiges Ziel:

[1, 2,

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

Langfristiges Ziel:

Stärkung von Prologs puren und monotonen Elementen

1, 2, 3,

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

Langfristiges Ziel:

Stärkung von Prologs puren und monotonen Elementen

+ Programmierspracheneigenschaften ähnlich FP
+ constraintverträglich
+ verbesserte Modellierbarkeit (z.B. Terminationsanalysen)
+ verbesserte Erklärungen (*slices* statt *traces*)

1, 2, 3, 4,

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

Langfristiges Ziel:

Stärkung von Prologs puren und monotonen Elementen

+ Programmierspracheneigenschaften ähnlich FP
+ constraintverträglich
+ verbesserte Modellierbarkeit (z.B. Terminationsanalysen)
+ verbesserte Erklärungen (*slices* statt *traces*)
+ ISO/IEC 13211 Normarbeit in SC 22 WG 17 über DIN, ON

Bisher:

1, 2, 3, 4, 5,

Lambdas und Schleifen in monotonen Logikprogrammen

Ulrich Neumerkel, TU Wien

Langfristiges Ziel:

Stärkung von Prologs puren und monotonen Elementen

+ Programmierspracheneigenschaften ähnlich FP
+ constraintverträglich
+ verbesserte Modellierbarkeit (z.B. Terminationsanalysen)
+ verbesserte Erklärungen (*slices* statt *traces*)
+ ISO/IEC 13211 Normarbeit in SC 22 WG 17 über DIN, ON

Bisher:

Korrekte Unifikation, Arithmetik und Constraints, Termination

Seiteneffektfreie Ein-/Ausgabe

Umsetzung in SWI-Prolog, z.T. YAP-Prolog, z.t.T. B-Prolog

1, 2, 3, 4, 5, 6

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  - Variablengültigkeit problematisch, Monotonieverletzungen
  - oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  - oft inkompatibel mit Constraints
  - λ in LP bisher *littera non grata*
- Jetzt:

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  - Variablengültigkeit problematisch, Monotonieverletzungen
  - oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  - oft inkompatibel mit Constraints
  - λ in LP bisher *littera non grata*
- Jetzt: $\lambda \ = \ \lambda$

1, 2,

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  - Variablengültigkeit problematisch, Monotonieverletzungen
  - oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  - oft inkompatibel mit Constraints
  - λ in LP bisher *littera non grata*
- Jetzt: $\lambda \; = \; \backslash$

1, 2, 3,

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  - Variablengültigkeit problematisch, Monotonieverletzungen
  - oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  - oft inkompatibel mit Constraints
  - λ in LP bisher *littera non grata*
- Jetzt: $\lambda \ = \ \backslash\!\wedge$

1, 2, 3, 4,

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  - Variablengültigkeit problematisch, Monotonieverletzungen
  - oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  - oft inkompatibel mit Constraints
  - λ in LP bisher *littera non grata*
- Jetzt: $\lambda \ = \ \backslash\wedge$

1, 2, 3, 4, 5,

Lambda-Ausdrücke

- bisher nur mit speziellen Erweiterungen, übersetzergesteuert
  – Variablengültigkeit problematisch, Monotonieverletzungen
  – oft Probleme bei call/1: ..., Ziel, ... $\Longleftrightarrow$ ..., call(Ziel), ...
  – oft inkompatibel mit Constraints
  – λ in LP bisher *littera non grata*
- Jetzt: $\lambda \;=\; \backslash \;\; \wedge$
  $\wedge$ ... Parameterübergabe
  $\backslash$ ... Umbenennung

\+ als minimale Bibliothek umgesetzt, reines ISO-Prolog
\+ Variablengültigkeit unterschiedlich deklariert
ohne Deklaration lokal/gebunden
globale/freie Variable: *Vars* +\ Ziel
\+ Übersetzer belässt Variablengültigkeit
verbessert Ressourcenverbrauch
erkennt Fehler

1, 2, 3, 4, 5, 6

library(lambda) — Umbenennung

```
?- X+Y = 1+2.
X = 1,
Y = 2.
```

```
?- \ ( X+Y = 1+2 ).
true.
```

```
?- X+\ ( X+Y = 1+2 ).
X = 1.
```

```
?- [X,Y]+\ ( X+Y = 1+2 ).
X = 1.
Y = 2
```

```
?- dif(X,Y).
dif(X,Y).
```

```
?- \ dif(X,Y).
dif(_1,_2).
```

```
?- X+\ dif(X,Y).
dif(X,_1).
```

```
?- [X,Y]+\ dif(X,Y).
dif(X,Y).
```

library(lambda) — call/N, Parameterübergabe, λ

```
?- call(dif(X), Z).
dif(X, Z).
```

```
?- call(Y^dif(X,Y),Z).
Y = Z,
dif(X, Z).
```

```
?- call(Y^dif(X,Y),Z1), call(Y^dif(X,Y),Z2).
Y = Z2,
Z1 = Z2,              %!!!!  Fehlbindung
dif(X, Z2), dif(X, Z2).
```

λ = Umbenennung + Parameterübergabe

```
?- call(X+\Y^dif(X,Y),Z1), call(X+\Y^dif(X,Y),Z2).
dif(X, Z2), dif(X, Z1).
```

```
?- maplist(X+\Y^dif(X,Y), [Z1,Z2]).
dif(X, Z2), dif(X, Z1).
```

`library(lambda)`

- Passt in bestehende ISO Norm
- Normerweiterung überflüssig
- http://www.complang.tuwien.ac.at/ulrich/Prolog-inedit/lambda.pl

für SWI, YAP mit Constraints