

Tutorium 09: Typinferenz

Paul Brinkmeier

16. Dezember 2019

Tutorium Programmierparadigmen am KIT

Heutiges Programm

- Keine ÜBs :(
- Unifikation in Prolog
- Typinferenz mit Typisierungsbäumen
- Allgemeinsten Unifikator

Datentypen in Prolog

```
a(b, c, d).  
defg.  
bintree(bintree(1, 2), bintree(3, bintree(4, 5))).  
list(cons(1, cons(2, cons(3, nil)))).  
'Abcd'('X', 'Y', 'Z').
```

- Funktor \approx Name + Liste von Prolog-Ausdrücken
- Liste leer \rightsquigarrow „Atom“
- Name wird immer klein geschrieben
 - Großbuchstaben: bspw. 'List'

```
?- X = pumpkin.  
?- Y = honey_bunny.  
?- Z = vincent.  
?- [A, B, C] = [1, 2, 3].  
?- f(L, rechts) = f(links, R)
```

- Variablen werden immer groß geschrieben
- = ist nicht Zuweisung, sondern Unifikation
- Unifikation \approx (formales) Pattern-Matching

Unifikation zweier Prolog-Terme nach Robinson

```
unify(lhs, rhs) =  
  if lhs == rhs: return []  
  if isVar(lhs) and varName(lhs) not in fv(rhs):  
    return [name(lhs) => rhs]  
  if isVar(rhs) and varName(rhs) not in fv(lhs):  
    return [name(rhs) => lhs]  
  if isFunctor(lhs) and isFunctor(rhs)  
  and functorName(lhs) == functorName(rhs):  
  and functorLen(lhs) == functorLen(rhs):  
    // unify functorArgs(lhs) and functorArgs(rhs)  
    // concatenate all unifiers  
  throw error
```

Unifikation zweier Funktoren nach Robinson

```
unifier = []  
for i in 0..functorLen(lhs):  
    unifier.addAll(unify(  
        unifier.apply(lhs.getArg(i)),  
        unifier.apply(rhs.getArg(i))  
    ))  
  
return unifier
```

- Argumente des linken und rechten Funktors werden nacheinander unifiziert
- Dabei müssen die vorherigen Substitutionen beachtet werden

Unifikation zweier Funktoren nach Robinson

```
unifier = []  
for i in 0..functorLen(lhs):  
    unifier.addAll(unify(  
        unifier.apply(lhs.getArg(i)),  
        unifier.apply(rhs.getArg(i))  
    ))  
  
return unifier
```

- Argumente des linken und rechten Funktors werden nacheinander unifiziert
- Dabei müssen die vorherigen Substitutionen beachtet werden
- Umformung des Robinson-Algorithmus der Vorlesung, nur zur Veranschaulichung!

Unifiziert:

- $A = x$
- $B = f(x)$
- $C = g(C)$
- $f(x, A, z) = f(x, y, B)$
- $g(x, A, z) = f(x, A, A)$
- $f(g(z)) = f(D)$

Ergebnis: Entweder **fail** oder ein Unifikator.

Typinferenz

Ein Term im λ -Kalkül hat eine der drei folgenden Formen:

Notation	Besteht aus	Bezeichnung
x	x : Variablenname	Variable
$\lambda p.b$	p : Variablenname b : λ -Term	Abstraktion
$f\ a$	f, a : λ -Terme	Funktionsanwendung

Wiederholung

- Bisher: Typisierung *prüfen*
- Gegeben Term t , Typ τ und Kontext Γ , zeige, dass $\Gamma \vdash t : \tau$
 - „ t hat Typ τ im Kontext Γ “

$$\frac{\dots}{f : \text{int} \rightarrow \beta \vdash \lambda x.f \ x : \text{int} \rightarrow \beta} \text{ABS}$$

- „Zeige, dass $\lambda x.f \ x$ den Typ $\text{int} \rightarrow \beta$ hat“

Wiederholung

- Bisher: Typisierung *prüfen*
- Gegeben Term t , Typ τ und Kontext Γ , zeige, dass $\Gamma \vdash t : \tau$
 - „ t hat Typ τ im Kontext Γ “

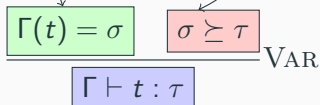
$$\frac{\dots}{f : \text{int} \rightarrow \beta \vdash \lambda x.f \ x : \text{int} \rightarrow \beta} \text{ABS}$$

- „Zeige, dass $\lambda x.f \ x$ den Typ $\text{int} \rightarrow \beta$ hat“
- Jetzt drehen wir den Spieß um:

$$\frac{\dots}{f : \text{int} \rightarrow \beta \vdash \lambda x.f \ x : \alpha_1} \text{ABS}$$

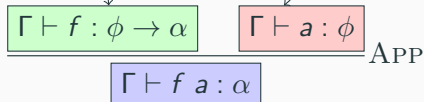
- „Finde den allgemeinsten Typen α_1 von $\lambda x.f \ x$ “

- „Der Typkontext Γ enthält einen Typ σ für t “
- „ σ kann mit τ instanziiert werden“



- dann gilt:
- „Variable t hat im Kontext Γ den Typ τ “
- $\sigma \succeq \tau \rightsquigarrow$ „ σ hat τ s Struktur und ist (mind.) allgemeiner“
 - $\text{int} \rightarrow \text{int} \succeq \text{int} \rightarrow \text{int}$
 - $\forall \alpha. \alpha \rightarrow \alpha \succeq \text{int} \rightarrow \text{int}$
 - $\alpha \rightarrow \alpha \not\succeq \text{int} \rightarrow \text{int}$
 - $\text{int} \rightarrow \text{int} \not\succeq \forall \alpha. \alpha \rightarrow \alpha$

- „ f ist im Kontext Γ eine Funktion, die ϕ s auf α s abbildet“
- „ a ist im Kontext Γ ein Term des Typs ϕ “



- dann gilt:
- „ a eingesetzt in f ergibt einen Term des Typs α “

- „Damit b als Funktion von p typisierbar ist...“
- „... müssen wir den Typ von p in den Kontext einfügen“

$$\frac{\boxed{\Gamma, p : \pi} \quad \boxed{\vdash b : \rho}}{\boxed{\Gamma \vdash \lambda p. b : \pi \rightarrow \rho}} \text{ABS}$$

- dann gilt:
- „ $\lambda p. b$ ist eine Funktion, die π s auf ρ s abbildet“

Vorgehensweise zur Typinferenz:

- Stelle Typherleitungsbaum auf
 - In jedem Schritt werden neue Typvariablen α_i angelegt
 - Statt die Typen direkt im Baum einzutragen, werden Gleichungen in einem Constraint-System eingetragen
- Unifiziere Constraint-System zu einem Unifikator
 - Robinson-Algorithmus, im Grunde wie bei Prolog

Robinson-Algorithmus

```
unify [] = []
unify [lhs = rhs | rest] =
    if lhs == rhs then unify rest
    if (lhs == Var a) and a not in fv(rhs):
        unify (apply [a => rhs] rest) ++ [a => rhs]
    if (rhs == Var a) and a not in fv(lhs):
        unify (apply [a => lhs] rest) ++ [a => lhs]
    if (lhs == a -> b) and (rhs == c -> d):
        unify (rest ++ [a = c, b = d])
    otherwise:
        fail
```

Erzeugt Unifikator zu einem Constraint-System.

Typinferenz: Übungsaufgabe

$$\frac{\dots}{f : \text{int} \rightarrow \beta \vdash \lambda x.f \ x : \alpha_1} \text{ABS}$$

- „Finde den allgemeinsten Typen α_1 von $\lambda x.f \ x$ “

Erinnerung:

- Baum mit durchnummerierten α_i aufstellen
- Constraints sammeln:

$$\frac{\Gamma(x) = \sigma \quad \sigma \succeq \tau}{\Gamma \vdash x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash f : \xi \quad \Gamma \vdash x : \phi}{\Gamma \vdash f \ x : \alpha} \text{APP}$$

$$\frac{\Gamma, p : \pi \vdash b : \beta}{\Gamma \vdash \lambda p.b : \alpha} \text{ABS}$$

Constraint: $\{\sigma = \tau\}$

Constraint: $\{\xi = \phi \rightarrow \alpha\}$

Constraint: $\{\alpha = \pi \rightarrow \beta\}$

- Constraint-System auflösen

Typinferenz: Übungsaufgabe

$$\frac{\dots}{\vdash \lambda x. \lambda f. f \ x \ x : \alpha_1} \text{ABS}$$

- „Finde den allgemeinsten Typen α_1 von $\lambda x. \lambda f. f \ x \ x$ “

Erinnerung:

- Baum mit durchnummerierten α_i aufstellen
- Constraints sammeln:

$$\frac{\Gamma(x) = \sigma \quad \sigma \succeq \tau}{\Gamma \vdash x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash f : \xi \quad \Gamma \vdash x : \phi}{\Gamma \vdash f \ x : \alpha} \text{APP}$$

$$\frac{\Gamma, p : \pi \vdash b : \beta}{\Gamma \vdash \lambda p. b : \alpha} \text{ABS}$$

Constraint: $\{\sigma = \tau\}$

Constraint: $\{\xi = \phi \rightarrow \alpha\}$

Constraint: $\{\alpha = \pi \rightarrow \beta\}$

- Constraint-System auflösen

Gute Ferienzeit!