

# Merkzettel für „Theorie der Programmierung“

Marco Ammon

5. Oktober 2018

## Termersetzungssysteme

### Terminierung

*Satz 2.31 (15)*

Sei  $>$  Reduktionsordnung, und es gelte:  $\forall t, s. t \rightarrow_0 s \Rightarrow t > s$ . Dann ist  $\rightarrow$  stark normalisierend.

### Polynomordnungen

*Kor. 2.44 (18)*

Wir definieren  $A \subseteq \mathbb{N}$  und für jede  $n$ -stellige Operation  $f$  ein Polynom  $p_f(x_1, \dots, x_n)$ . Wenn die linken Seiten der Umformungsregeln  $>_A$  den rechten sind, ist das zugehörige TES stark normalisierend.

### Konfluenz

#### Critical Pairs

- Definition kritisches Paar:  
Seien  $l_1 \rightarrow_0 r_1$  und  $l_2 \rightarrow r_2$  zwei Umformungsregeln des TES sowie  $FV(l_1) \cap FV(l_2) = \emptyset$  (ggf. nach Umbenennung). Sei  $l_1 = C(t)$ , wobei  $t$  nicht nur eine Variable ist, so dass  $t$  und  $l_2$  unifizierbar sind. Sei  $\sigma = mgu(t, l_2)$ . Dann heißt  $(r_1\sigma, C(r_2)\sigma)$  ein kritisches Paar.

*Def. 2.55 (22)*

- Ein TES  $T$  ist genau dann lokal konfluent, wenn in  $T$  alle kritischen Paare zusammenführbar sind.

*Satz 2.60 (24)*

- Ein stark normalisierendes und lokal konfluentes TES ist konfluent.

*Satz 2.51 (21)*

## $\lambda$ -Kalkül

### Ungetypt

- $\beta$ -Reduktion

$$(\lambda x. t)s \rightarrow_\beta t[s/x]$$

- $\eta$ -Reduktion

$$\lambda x. yx \rightarrow_\eta y$$

## Auswertungsstrategien

- applikativ (*leftmost-innermost*)  $\rightarrow_a$  Def. 3.13 (33)
  - $\lambda x.t \rightarrow_a \lambda x.t'$ , wenn  $t \rightarrow_a t'$
  - $ts \rightarrow_a t's$ , wenn  $t \rightarrow_a t'$
  - $ts \rightarrow_a ts'$ , wenn  $s \rightarrow_a s'$  und  $t$  normal ist
  - $(\lambda x.t)s \rightarrow_a t[s/x]$ , wenn  $t$  und  $s$  normal sind
  - effizient
- normal (*leftmost-outermost*)  $\rightarrow_n$  Def. 3.14 (34)
  - $(\lambda x.t)s \rightarrow_n t[s/x]$
  - $\lambda x.t \rightarrow_n \lambda x.t'$ , wenn  $t \rightarrow_n t'$
  - $ts \rightarrow_n t's$  wenn  $t \rightarrow_n t'$  und  $t$  keine  $\lambda$ -Abstraktion ist
  - $ts \rightarrow_n ts'$ , wenn  $s \rightarrow_n s'$  und  $t$  normal und keine  $\lambda$ -Abstraktion ist
  - terminiert immer, falls Normalform existiert (nach Standardisierungssatz) Satz 3.17 (35)

## Einfach getypt ( $\lambda \rightarrow$ )

- Church: Annotation der Variablen mit Typen, nur herleitbare Terme hinschreibbar S. 39
- Curry: Alle Terme hinschreibbar, dann Aussondern der nicht typisierbaren
- Typregeln: S. 39

$$\begin{aligned}
 (Ax) \quad & \frac{}{\Gamma \vdash x : \alpha} \quad x : \alpha \in \Gamma \\
 (\rightarrow_e) \quad & \frac{\Gamma \vdash t : \alpha \rightarrow \beta \quad \Gamma \vdash s : \alpha}{\Gamma \vdash ts : \beta} \\
 (\rightarrow_i) \quad & \frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \rightarrow \beta}
 \end{aligned}$$

- Typisierungsprobleme S. 40
  - Typcheck: „Gilt  $\Gamma \vdash t : \alpha$ ?“
  - Typinferenz: „Was ist das beste  $\alpha$  / Existiert  $\alpha$  mit  $\Gamma \vdash t : \alpha$ ?“
  - Type inhabitation: „Existiert  $t$  mit  $\Gamma t : \alpha$ ?“
- Inversionslemma Lem. 3.29 (41)
  1.  $\Gamma \vdash x : \alpha \Rightarrow x : \alpha \in \Gamma$
  2.  $\Gamma \vdash ts : \beta \Rightarrow \exists \alpha$  mit  $\Gamma \vdash t : \alpha \rightarrow \beta$  und  $\Gamma \vdash s : \alpha$
  3.  $\Gamma \vdash \lambda x.t : \gamma \Rightarrow \gamma = \alpha \rightarrow \beta$  mit  $\Gamma, x : \alpha \vdash t : \beta$
- Typinferenz S. 41
  - Typsubstitution  $\sigma$  ist Lösung von  $\Gamma \vdash t : \alpha$ , wenn  $\Gamma \sigma \vdash t : \alpha \sigma$  herleitbar
  - Substitutionen:  $\sigma_1$  allgemeiner als  $\sigma_2 \Leftrightarrow \exists \tau. \sigma_1 \tau = \sigma_2$  GLoIn, S. 38
  - Prinzipaltyp von  $\Gamma, t$  ist  $\sigma(a)$  für allgemeinste Lösung  $\sigma$  von  $\Gamma \vdash t : a$  ( $a$  frisch)
  - Algorithmus W (Hindley/Milner) Alg. 3.31 (42)

\* Menge  $PT$  von Typgleichungen

$$PT(\Gamma; x; \alpha) = \{a \doteq b \mid x : \beta \in \Gamma\}$$

$$PT(\Gamma; \lambda x.t; \alpha) = PT((\Gamma; x : a); t; b) \cup \{a \rightarrow b \doteq \alpha\} \text{ mit } a, b \text{ frisch}$$

$$PT(\Gamma; ts; \alpha) = PT(\Gamma; t; a \rightarrow \alpha) \cup PT(\Gamma; s; a) \text{ mit } a \text{ frisch}$$

\* Typinferenz des Terms  $u$  mit leerem Kontext:

$$\varepsilon := PT(\emptyset; u; a)$$

$$\Rightarrow \text{Prinzipaltyp von } u: \text{mgu}(\varepsilon)(a)$$

- Subjektreduktion: Wenn  $\Gamma \vdash t : \alpha$  und  $t \rightarrow_{\beta}^* s$ , dann auch  $\Gamma \vdash s : \alpha$ , aber nicht umgekehrt!

Satz 3.38 (45)

## Induktive Datentypen

### Mengenkonstruktionen

Def. 4.12 (56)

$$X_1 \times X_2 = \{(x_1, x_2) \mid x_i \in X_i \text{ für } i = 1, 2\} \quad (\text{„struct“})$$

$$X_1 + X_2 = \{(i, x) \mid i = 1, 2, x \in X_i\} \quad (\text{„union“})$$

$$1 = \{*\} \quad \text{„()“ in Haskell}$$

Lem. 4.13 (57)

Seien  $f_i : X_i \rightarrow Y_i$ ,  $g_i : X_i \rightarrow Z$  und  $h_i : Z \rightarrow X_i$  mit  $i \in \{1, 2\}$ .

$$f_1 \times f_2 : X_1 \times X_2 \rightarrow Y_1 \times Y_2, \quad (f_1 \times f_2)(x_1, x_2) = (f_1(x_1), f_2(x_2))$$

$$f_1 + f_2 : X_1 + X_2 \rightarrow Y_1 + Y_2, \quad (f_1 + f_2)(i, x) = (i, f_i(x))$$

$$[g_1, g_2] : X_1 + X_2 \rightarrow Z, \quad [g_1, g_2](i, x) = g_i(x)$$

$$\langle h_1, h_2 \rangle : Z \rightarrow X_1 \times X_2, \quad \langle h_1, h_2 \rangle(z) = (h_1(z), h_2(z))$$

$$in_i : X_i \rightarrow X_1 + X_2, \quad in_i(x) = (i, x)$$

$$\pi_i : X_1 \times X_2 \rightarrow X_i, \quad \pi_i(x_1, x_2) = x_i$$

$$1 : 1 \rightarrow 1$$

Es gilt

$$[g_1, g_2] \circ in_i = g_i$$

$$f_1 + f_2 = [in_1 \circ f_1, in_2 \circ f_2]$$

$$[r \circ in_1, r \circ in_2] = r \text{ für } r : X_1 + X_2 \rightarrow Z$$

$$\pi_i \circ \langle h_1, h_2 \rangle = h_i$$

$$f_1 \times f_2 = \langle f_1 \circ \pi_1, f_2 \circ \pi_2 \rangle$$

$$\langle \pi_i \circ f, \pi_2 \circ f \rangle = f \text{ für } f : Z \rightarrow X_1 \times X_2$$

### Strukturelle Induktion

- über einsortige Datentypen

S. 63

- Induktionsanfang: „Anfangs“-Konstruktor (etwa *Nil*)
- Induktionsschritt: alle anderen Konstruktoren (etwa *cons*)

- über mehrsortige Datentypen

S. 64

- Funktionen müssen immer auf allen Datentypen definiert werden

## Kodatentypen

- Definition über „Destruktoren“ etwa  $hd$  und  $tl$

## Koinduktion

- Bisimulation  $R \subseteq A^\omega \times A^\omega$ , wenn für alle  $(s, t) \in R$  gilt:

Def. 4.39 (74)

$$\begin{aligned} hd\ s &= hd\ t \\ (tl\ s) &R\ (tl\ t) \end{aligned}$$

- Wenn  $R$  eine Bisimulation ist, gilt  $sRt \Rightarrow s = t$

Satz 4.40 (74)

## Kodatentypen mit Alternativen

## System F

### Curry

- Typen:

Def. 5.1 (84)

$$\alpha, \beta := a \mid \alpha \rightarrow \beta \mid \forall a. \alpha \quad (a \in V)$$

- Typisierung:

Def. 5.1 (84)

$$\begin{aligned} (Ax) \quad & \frac{}{\Gamma \vdash x : \alpha} (x : \alpha \in \Gamma) \\ (\rightarrow_e) \quad & \frac{\Gamma \vdash t : \alpha \rightarrow \beta \quad \Gamma \vdash s : \alpha}{\Gamma \vdash ts : \beta} \\ (\rightarrow_i) \quad & \frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x. t : \alpha \rightarrow \beta} \\ (\forall_i) \quad & \frac{\Gamma \vdash s : \alpha \quad a \notin FV(\Gamma)}{\Gamma \vdash s : \forall a. \alpha} \\ (\forall_e) \quad & \frac{\Gamma \vdash s : \forall a. \alpha}{\Gamma \vdash s : (\alpha[\beta/a])} \end{aligned}$$

## Church-Kodierung

S. 84

- Natürliche Zahlen

$$\begin{aligned} \mathbb{N} &:= \forall a. (a \rightarrow a) \rightarrow a \rightarrow a \\ zero &: \mathbb{N} \\ zero &= \lambda f x. x \\ suc &: \mathbb{N} \rightarrow \mathbb{N} \\ suc &= \lambda n f x. f(n f x) \\ fold &: \forall a. (a \rightarrow a) \rightarrow a \rightarrow \mathbb{N} \rightarrow a \\ fold &= \lambda f x n. n f x \\ add &: \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N} \\ add &= \lambda n. fold\ suc\ n \end{aligned}$$

- Paare

$$\begin{aligned}
(a \times b) &:= \forall r. (a \rightarrow b \rightarrow r) \rightarrow r \\
pair &: \forall ab. a \rightarrow b \rightarrow (a \times b) \\
pair &= \lambda xyf. fxy \\
fst &: \forall ab. (a \times b) \rightarrow a \\
fst &= \lambda p. p(\lambda xy. x) \\
snd &: \forall ab. (a \times b) \rightarrow b \\
snd &= \lambda p. p(\lambda xy. y)
\end{aligned}$$

- Summen

$$\begin{aligned}
(a + b) &:= \forall r. (a \rightarrow r) \rightarrow (b \rightarrow r) \rightarrow r \\
inl &: \forall ab. a \rightarrow (a + b) \\
inl &= \lambda xfg. fx \\
inr &: \forall ab. b \rightarrow (a + b) \\
inr &= \lambda yfg. gy \\
case &: \forall abs. (a \rightarrow s) \rightarrow (b \rightarrow s) \rightarrow (a + b) \rightarrow s \\
case &= \lambda fgs. sf g
\end{aligned}$$

- Listen

$$\begin{aligned}
List\ a &:= \forall r. r \rightarrow (a \rightarrow r \rightarrow r) \rightarrow r \\
Nil &: \forall a. List\ a \\
Nil &= \lambda uf. u \\
Cons &: \forall a. a \rightarrow List\ a \rightarrow List\ a \\
Cons &= \lambda xlu f. fx(lu f) \\
len &: \forall a. List\ a \rightarrow \mathbb{N} \\
len &= \lambda l. l\ zero\ (\lambda xr. suc\ r)
\end{aligned}$$

## ML-Polymorphie

- Einschränkung von System F durch  $\forall$  nur auf oberster Ebene sowie Mehrfachinstanziierung polymorpher Funktionen nur in *let*-Konstrukt

S. 88

- Typen

$$\alpha, \beta := a \mid \alpha \rightarrow \beta$$

- Typschemata

$$S := \forall a_1, \dots, a_k. \alpha \quad (k \geq 0)$$

- Terme

$$t, s := x \mid t\ s \mid \lambda x. t \mid \text{let } x = t \text{ in } s$$

- Kontexte

$$\begin{aligned}
\Gamma &= (x_1 : S_1, \dots, x_n : S_n) \\
Cl(\Gamma, \alpha) &= \forall a_1, \dots, a_k. \alpha \quad \text{für } FV(\alpha) \setminus FV(\Gamma) = \{a_1, \dots, a_k\}
\end{aligned}$$

- Typisierungsregeln

S. 88

$$(\forall_e) \frac{}{\Gamma \vdash x : \alpha[\beta_1/a_1, \dots, \beta_k/a_k]} (x : \forall a_1, \dots, \forall a_k. \alpha) \in \Gamma$$

$$(\rightarrow_i) \frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x. t : \alpha \rightarrow \beta}$$

$$(\rightarrow_e) \frac{\Gamma \vdash t : \alpha \rightarrow \beta \quad \Gamma \vdash s : \alpha}{\Gamma \vdash ts : \beta}$$

$$(\text{let}) \frac{\Gamma \vdash t : \alpha \quad \Gamma, x : Cl(\Gamma, \alpha) \vdash s : \beta}{\Gamma \vdash \text{let } x = t \text{ in } s : \beta}$$

- Inversionslemma

S. 89

1. Wenn  $\Gamma \vdash x : \alpha$ , dann existieren Typen  $\beta_i$  und ein Typschema  $S = \forall a_1 \dots \forall a_k. \gamma$ , so dass  $(x : S) \in \Gamma$  und  $\alpha = \gamma[\beta_1/a_1, \dots, \beta_k/a_k]$
2. Wenn  $\Gamma \vdash (\text{let } x = s \text{ in } t) : \alpha$ , dann existiert ein Typ  $\beta$  mit  $\Gamma \vdash s : \beta$  und  $\Gamma, x : Cl(\Gamma, \beta) \vdash t : \alpha$

- erweiterter Algorithmus W mit Menge  $PT$  von Typgleichungen

$$PT(\Gamma; x; \alpha) = \{a \doteq \gamma[a'_1/a_1, \dots, a'_k/a_k] \mid (x : \forall a_1, \dots, \forall a_k. \gamma) \in \Gamma\}$$

$$PT(\Gamma; \lambda x. t; \alpha) = PT((\Gamma; x : a); t; b) \cup \{a \rightarrow b \doteq \alpha\} \text{ mit } a, b \text{ frisch}$$

$$PT(\Gamma; ts; \alpha) = PT(\Gamma; t; a \rightarrow \alpha) \cup PT(\Gamma; s; a) \text{ mit } a \text{ frisch}$$

$$PT(\Gamma; (\text{let } x = s \text{ in } t); \alpha) = PT(\Gamma\sigma, x : Cl(\Gamma\sigma, \sigma(b)); t; \alpha\sigma)$$

wobei  $\sigma = mgu(PT(\Gamma; s; b))$  mit  $b$  frisch

## Minimierung von deterministischen endlichen Automaten

1. Entferne aus  $Q$  alle nicht erreichbaren Zustände
2. Initialisiere  $R$  auf  $\{(q_1, q_2) \mid q_1 \in F \Leftrightarrow q_2 \in F\}$
3. Suche ein Paar  $(q_1, q_2) \in R$  und einen Buchstaben  $a \in \Sigma$  mit

$$(\delta(a, q_1), \delta(a, q_2)) \notin R$$

Wenn kein solches Paar gefunden wird, gehe zu Schritt 4. Andernfalls entferne  $(q_1, q_2)$  aus  $R$  und fahre bei 3. fort.

4. Identifiziere alle Zustandspaare in  $R$ .

## Unifikationsalgorithmus (Martelli/Montanari)

$S \cup \{x \doteq x\}$	$\rightarrow S$	(delete)
$S \cup \{f(E_1, \dots, E_n) \doteq f(D_1, \dots, D_n)\}$	$\rightarrow S \cup \{E_1 \doteq D_1, \dots, E_n \doteq D_n\}$	(decomp)
$S \cup \{f(E_1, \dots, E_n) \doteq g(D_1, \dots, D_k)\}$	$\rightarrow \perp$ (für $f \neq g$ )	(conflict)
$S \cup \{E \doteq x\}$	$\rightarrow S \cup \{x \doteq E\}$ (für $E$ keine Variable)	(orient)
$S \cup \{x \doteq E\}$	$\rightarrow \begin{cases} \perp & \text{(für } x \in FV(E), x \neq E) \\ S[E/x] \cup \{x \doteq E\} & \text{(für } x \notin FV(E), x \in FV(S)) \end{cases}$	(occurs)/(eli)