Polynomordnung

1 In Funktionsschreibweise bringen

Infixnotation:

$$x \uparrow (y \uparrow z) \to_0 \; x \uparrow (y \downarrow y)$$

Funktionsschreibweise:

$$P_{\uparrow}(x, P_{\uparrow}(y, z)) \to_0 \; P_{\uparrow}(x, P_{\downarrow}(y, y))$$

2 Kontext ggf. kuerzen

$$P_{\uparrow}(x, P_{\uparrow}(y, z)) \to_0 \; P_{\uparrow}(x, P_{\downarrow}(y, y))$$

$$\cancel{P(x}, P_{\uparrow}(y, z)) \to_0 \; \cancel{P(x}, P_{\downarrow}(y, y)))$$

$$P_{\uparrow}(y, z) \to_0 \; P_{\downarrow}(y, y)$$

3 Polynom finden

$$P_{\uparrow}(y, z) \to_0 \; P_{\downarrow}(y, y)$$

Ein "+" zwischen die Parameter setzen und Multiplikator vor beide sodass gilt:

$$P_{\uparrow}(y, z) > \; P_{\downarrow}(y, y) \;\; \underline{\textbf{bzw:}} \;\; ay + bx > cy + dy$$

Hinweise:
- linke Seite ist syntaktisch echt grösser als die Rechte ist aussreichendes Kriterium also z.B.:

$$P_{\downarrow}(P_{\downarrow}(x, y)) > P_{\downarrow}(x, y)$$

- niemals Minuswerte
- niemals '0' als Multiplikator

hier:

$$ay + bz > cy + dy$$

Wir nehmen an dass wir 'y' hoch genug setzen damit bx irrelevant wird:

$$ay > cy + dy = a > c + d$$

das ist nun trivial, wir raten:

$c = 1$

$d = 1$

$a = c + d + 1 = 3$

und damit die Polynome:

$$P \downarrow = x_1 + x_2 \quad und \quad P \uparrow = 3x_1 + x_2$$

4 Domänen und Grenzfälle

Unsere Polynome gelten potentiell für kleine Werte nicht, hier, nur für die 0 nicht, daher geben wir als Domäne an:

$$A = N \setminus \{0\}$$

und Funktionsdomäne:

$$\mathcal{A} = \{P_{\Downarrow}(), P_{\Uparrow()}\}$$

Wichtige Lemmas

Newman's Lemma:

Ein stark normalisierendes und lokal Konfluentes Termersetzungssystem (TES) ist konfluent.

Critical Pair Lemma:

Ein TES ist lokal konfluent, wenn alle kritischen Paare zusammenführbar sind.

Allgemeines Vorgehen:

Alle Regeln müssen gegen alle anderen gematched werden um lokale Konfluenz zu zeigen, für das Gegenteil reicht also logischerweise **ein** nicht- zusammenfürbares Paar.

Für Regelpaar (x)(y):
$\rightarrow$ l_1 = linke Seite von x
$\rightarrow$ l_2 = linke Seite von y (falls gleiche Variablennamen selbige umbennen)
$\rightarrow$ l_1 so substituieren, dass Regel y angewendet werden kann (aka MGU finden)
$\rightarrow$ l_1 sollte der MGU = l_2 sein $\rightarrow$ triviales Paar
$\rightarrow$ beide Regeln 1x anwenden und sehen ob man die entstehenden Terme wieder zusammen bringen kann *(durch Anwendung beliebiger Regeln des TES)*

Triviale Paare muessen nicht gezeigt werden. Ein Paar kann bereits nach Anwendung beider Regeln wieder gleich sein (z.B. $\neg\neg\neg x$). So ein Paar ist automatisch zusammenführbar, aber dennoch der Definition nach **<u>kein</u>** triviales Paar.

Beispiel Lokale Konfluenz zeigen:

Formeln:

$$x \Uparrow (y \Uparrow z) \rightarrow_0 x \Uparrow (y \Downarrow y) \quad (1)$$
$$x \Downarrow (x \Downarrow y) \rightarrow_0 x \Downarrow y \quad (2)$$

(1)(1)

$$l_1 = x \Uparrow (y \Uparrow z)$$
$$l_2 = u \Uparrow (v \Uparrow w)$$

damit:

$$mgu = [\ y \mapsto u\ ,\ z \mapsto (\ u\ \Uparrow\ w\ )\ ]$$

und die mit dem MGU substituierte Seite l_1:

$$l_1\sigma = x \Uparrow (\, u \ \Uparrow \ (\, v \Uparrow \ w\,))$$

$$l_1\sigma(1) = x \ \Uparrow \ (\, u \ \Downarrow \ u\,) \qquad\qquad \underbrace{l_1\sigma(1) = x \ \Uparrow \ (\, u \ \Uparrow \ \underbrace{(v \ \ \Downarrow \ v\,)}_{z'}}_{Regel\ (1)}$$

$$l_1\sigma(1) = x \ \Uparrow \ (\, u \ \Downarrow \ u\,) \qquad == \qquad l_1\sigma(1) = x \ \Uparrow \ (\, u \ \Downarrow \ u\,)$$

Analog mit (2)(2) und allen anderen kritischen Paaren (wobei (1)(2) und (2)(1) hier triviale kritische Paare wären!)

1 Generelles

1.1 Normale-/Lazy-Reduktion

- pre-order durch Baum ("von unten nach oben auswerten")
- leftmost-outermost
- Argumente zum Schluss auswerten

$$\begin{aligned} pow(a, pow(c, b)) &\ mit\ linkem\ pow\ anfangen \\ &\ dann\ a \\ &\ dann\ rechtes\ pow \\ &\ dann\ c \\ &\ dann\ d \end{aligned}$$

1.2 Applikative Reduktion:

- post-order durch Baum ("von oben nach unten")
- leftmost-innermost
- als erstes die Argumente auswerten

$$\begin{aligned} pow(a, pow(c, b)) &\ mit\ c\ anfangen \\ &\ dann\ b \\ &\ dann\ a \\ &\ dann\ rechtes\ pow \\ &\ dann\ linkes\ pow \end{aligned}$$

1.3 How to work with Lambda

- Buchstabe hinter λ wegnehmen
- Term von der respektiven Stelle hinten entfernen
- alle Vorkommen des Buchstabens mit dem Term ersetzen
- idealerweise bei mehreren λ in einem Term **immer** verschiedene Buchstaben verwenden
- wenn nicht genug Terme zum Auflösen von λ vorhanden sind, nicht damit anfangen, also z.B. bei $pred\ \lambda\, f\ a\ .\ fa\ one$ nicht one für f einsetzen sondern *pred* auflösen

2 SS15Probeklausur Beispielaufgabe

1) Reduktion

a) Normal/Lazy

$$\begin{aligned}
\mathit{pow2\ three} &= \mathit{three}\ (\mathit{mult\ two})\ \mathit{one} \\
&= \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)(\, \mathit{mult\ two})\ \mathit{one} \\
&= \lambda a\, .\, (\mathit{mult\ two})\, ((\mathit{mult\ two})\, ((\mathit{mult\ two})\ a\,)\ \mathit{one} \\
&= (\mathit{mult\ two})\, ((\mathit{mult\ two})\, ((\mathit{mult\ two})\ \mathit{one}\,)
\end{aligned}$$

b) Aplikativer Reihenfolge

$$\begin{aligned}
\mathit{pow2\ three} &= \mathit{pow2}\ (\, \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)\,) \\
&= (\, \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)\,)(\mathit{mult\ two})\ \mathit{one} \\
&= (\, \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)\,)(\mathit{mult}\ \lambda g\, b\, .\, g\, (\, g\ b\,))\ \mathit{one} \\
&= (\, \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)\,)(\mathit{mult}\ \lambda g\, b\, .\, g\, (\, g\ b\,))\ (\lambda h\, c\, .\, h\ c)
\end{aligned}$$

kleine Anmerkung hierzu noch:

$$\underbrace{(\, \lambda f\, a\, .\, f\, (\, f\, (\, f\ a\,)\,)}_{\mathit{oberste\ Funktion}}\underbrace{(\mathit{mult}\ \lambda g\, b\, .\, g\, (\, g\ b\,))}_{\mathit{erstes\ Argument}}\quad\underbrace{(\lambda h\, c\, .\, h\ c)}_{\mathit{zweites\ argument}}$$

d.h. wir würden hier beim ersten Argument weitermachen und "mult" auflösen (beim zweiten gäbe es gerade auch gar nichts mehr zu machen)

2) Typherleitung

Typherleitung für:

$$\{\, \mathit{mult}\ :\ \mathrm{N} \to \mathrm{N} \to \mathrm{N},\ \mathit{one}\ :\ \mathrm{N},\ \mathit{two}\ \mathrm{N} \vdash\ \lambda n.\, n\, (\, \mathit{mult\ two}\,)\ \mathit{one}\ : \mathrm{N} \to \mathrm{N}\}$$

mit Sonderhinweis, dass die Chruch-Numerale in System F den folgenden Typ besitzen:

$$\mathrm{N}\ :=\ \forall a.(a\ \to\ a)\ \to\ a\ \to\ a$$

Herleitung:

$$\dfrac{\dfrac{\dfrac{\dfrac{(5)\ \Gamma\ \cup \{\, n : \mathrm{N}\, \} \vdash \overbrace{n\ :\ \forall a\, .\, (\, a\ \to\ a\,) \to (\, a\ \to\ a\,)}^{\mathit{per\ Definition} = \mathrm{N}} \qquad \overbrace{\mathit{mult} :\ \mathrm{N} \to \mathrm{N} \to \mathrm{N}}^{\mathit{Ax}} \qquad \overbrace{\mathit{two} :\ \mathrm{N}}^{\mathit{Ax}}}{(4)\ \Gamma\ \cup \{\, n : \mathrm{N}\, \} \vdash n\ :\ (\mathrm{N} \to \mathrm{N}) \to (\mathrm{N} \to \mathrm{N}) \qquad \mathit{mult\ two}\ \mathrm{N} \to \mathrm{N}}}{(3)\ \Gamma\ \cup \{\, n : \mathrm{N}\, \}\ \vdash\ n\, (\, \mathit{mult\ two}\,)\ : \mathrm{N} \to\ \mathrm{N} \qquad \mathit{one}\ : \mathrm{N}}}{(2)\ \Gamma\ \cup \{\, n : \mathrm{N}\, \}\ \vdash\ n\, (\, \mathit{mult\ two}\,)\ \mathit{one}\ : \mathrm{N}}}{(1)\ \{\underbrace{\mathit{mult}\ :\ \mathrm{N} \to \mathrm{N} \to \mathrm{N},\ \mathit{one}\ :\ \mathrm{N},\ \mathit{two}\ \mathrm{N}}_{:=\Gamma} \vdash\ \lambda n.\, n\, (\, \mathit{mult\ two}\,)\ \mathit{one}\ : \mathrm{N} \to \mathrm{N}}$$

Erklärungen:
(1) $\rightarrow_i$
(2)(3) $\rightarrow_e$
(3)(rechts)[one : N] mit Ax fertig
(4) $\forall_e$ (wir wissen, dass N : $(a- > a)- > a- > a$ ist, und dass {n : N ist}; damit n hinten also mit dem Kontex aufgeht muss es von der Form $(a- > a)- > a- > a$ sein)
(4)(links) $\forall_e$ (rechts) $\rightarrow_e$

Eine Übersicht der Regeln findet sich in den Übungsfolien oder dem SS14 Spickzettel!

1 Korekursion

Gegeben:

$$\begin{aligned}&codata Signal\ where\\&\quad currentSample : Signal \to Int\\&\quad discardSample : Signal \to Signal\end{aligned}$$

sowie:

$$\begin{aligned}&currentSample(flat\ x) = x && discardSample(flat\ x) = flat\ x\\&currentSample(square\ x\ y) = x && discardSample(square\ x\ y) = square\ y\ x\end{aligned}$$

Es soll gelten:
sampler t s = s , wenn $t > 0$ und 0 sonst
sowie insbesondere:

$$\begin{aligned}&sampler :\ Signal \to Signal \to Signal\\&sampler(square\ 0\ 1)(square\ x\ 0) = flat\ 0\\&sampler(square\ 1\ 0)(flat\ x) = square\ x\ 0\end{aligned}$$

Schritt 1:
→ sampler Funktion in codata-Funktionen einsetzen

$$\begin{aligned}&currentSample(sampler\ t\ s)\\&discardSample(sampler\ t\ s)\end{aligned}$$

Schritt 2:
→ Bedingung von oben bei erster Funktion anwenden also:

$$\begin{aligned}currentSample(sampler\ t\ s) = {}&if\ (currentSample\ \ t > 0)\\&then\ (currentSample\ \ s)\\&else\ 0\end{aligned}$$

sowie zweite Formel zum Aufteilen verwenden:

$$discardSample(sampler\ t\ s) = sampler(discardSample\ \ t)(discardSample\ \ s)$$

2 Ko-Induktion:

Induktionsanfang:
→ anhand erster Formel 'R' aufstellen

$$R = \{\underbrace{sampler(square\,0\,1)(square\,x\,0)}_{linke\ Seite}, \quad \underbrace{flat\,0}_{rechte\ Seite} \quad | \ \underbrace{x \in Int}_{von\ oben}\}$$

→ "R ist Bisimulation" hinschreiben, linke Seite aufloesen

$$currentSample(\ sampler(square\,0\,1)(square\,x\,0)) = 0$$

→ wenn hier am Ende kein Term rauskommt, dann koennen wir einfach die **rechte Seite bei der normalen Funktion** einsetzen und selbige aufloesen:

$$currentSample(flat\ 0) = 0$$

→ sehr gut, die rechten Seiten sind gleich wir sind hier also fertig

<u>WICHTIG</u> Jetzt das selbe noch mit discardSample()!

zweite Bedingung:

$$discardSample(sampler(square\ 1\ 0)(square\ 0\ x))$$
$$=sampler(discardSample(square\ 1\ 0))(discardSample(square\ 0\ x))$$

das ist doof denn:

$$discardSample(square\ x\ 0) = discardSample(square\ 0\ x)$$

Schritt 3:

$$R' = R \cup \{\underbrace{sampler(discardSample(square\ 1\ 0))(discardSample(square\ 0\ x))}_{aufgeloeste\ 2.\ Bedingung}, discardSample(square\ x\ 0$$

wir muessen jetzt zeigen:

$$\underbrace{currentSample(sampler(square\,1\,0)(square\,0\,x))}_{currentSample(\ x\ 0\ )} = \underbrace{currentSample(square\,0\,x)}_{selbes\ wie\ rechts}$$

passt also, jetzt wie oben auch zweite Funktion:

$$discardSample(sampler(square10)(square0x)) = \underbrace{sampler(square\,1\,0)(square\,0\,x)}_{wieder\ linke\ Seite\ in\ R'}$$

$$discardSample(\ square\ x\ 0\ ) = \underbrace{discardSample(\ square\ 0\ x\ )}_{wieder\ rechte\ Seite}$$

und fertig!

Sonstiges:

Eine Relation R $\subset$ A^{ω} x A^{ω} heisst Bisimulation, wenn für alle $(s, s') \in$ R gilt:
currentSample s = currentSample s'   weil Int zurück
(discardSample s) R (discardSample t)   weil wieder Signal

Wenn R eine Bisimulation ist, dann gilt R $\subseteq$ id , d.h.

$$sRt \Rightarrow s = t$$

für alle s,t $\in$ A^{ω}

Strukturelle Induktion

Referenzaufgabe:

data List a $=$ *Nil* $|$ *Cons* a **List** a

$$snoc\ Nil\ a = Cons\ a\ Nil$$
$$snoc(Cons\ x\ xs)\ a = Cons\ x\ (snoc\ xs\ a)$$

$$Nil + ys = ys$$
$$(Cons\ x\ xs) + ys = Cons\ x\ (xs + ys)$$

<u>Beweisen sie dass:</u>
$\forall e,\ xs,\ ys$
$xs + (Cons\ e\ ys) = (snoc\ xs\ e) + ys$

Induktionsanfang (IA):

$xs = Nil\ \Rightarrow$ Einsetzen und beide Seiten maximal vereinfachen

$$\begin{aligned} Nil + (Cons\ e\ ys) &= (snoc\ Nil\ e) + ys \\ Cons\ e\ ys &= (snoc\ Nil\ e) + ys \\ Cons\ e\ ys &= (ConseNil) + ys \\ Cons\ e\ ys &= Cons\ e(Nil + ys) \\ Cons\ e\ ys &= Cons\ e\ ys \end{aligned}$$

Induktionshypothese/-voraussetzeung (IH/IV):

- xs durch allgemeines as ersetzen

$$\forall\, e\, ys\ .\ as + (Cons\ e\ ys) = (snoc\ as\ e) + ys$$

Induktionsschritt (IS):

$xs \ = \ Cons\ a\ as$
$\Rightarrow$ Einsetzen und wieder beide Seiten maximal vereinfachen,auf einer Seite die Induktionshypothese reinfrikeln (idR. nur auf einer Seite,z.B. auf der linken)

$$\begin{aligned}
Cons\ a\ as\ + (Cons\ e\ ys) \ &= \ snoc\ ((Cons\ a\ as)\ e) + ys \\
Cons\ a\ (\underbrace{as\ + (Cons\ e\ ys)}_{IH\ links}) \ &= \ snoc\ ((Cons\ a\ as)\ e) + ys \\
Cons\ a\ (\underbrace{(snoc\ as\ e)\ + \ ys}_{IH\ rechts}) \ &= \ snoc\ ((Cons\ a\ as)\ e) + ys \\
Cons\ a\ ((snoc\ as\ e)\ + \ ys)) \ &= \ Cons\ a\ ((\ snoc\ as\ e)\ + \ ys))
\end{aligned}$$

Q.E.D. und fertig!

Pumping Lemma für Reguläre Sprachen

Konkret geht es in diesem Beispiel um eine Sprache die alle geöffneten Klammern auch wieder schließen soll, oder allgemeiner, um eine Sprache die sich eine Anzahl merken muss.

'L' ist nicht regulär wenn:

$$\forall l \geq 1 \,.\, \exists w \in L \; mit \; |w| \geq l$$

sodass:
∀ uvz
mit:
w=uvz
|v| ≥ 1
|uv| ≤ l
gilt:
uv^kz ∉ L

Beweis, dass 'L' <u>NICHT</u> regulär ist

Sei l ≥ 1 gegeben, wähle:

$$w \;=\; \underbrace{(\,(\,(\,...\,(}_{l+k\ mal} \qquad a \qquad \underbrace{,\,b\,)\,,b)\,...\,)}_{l+k\ mal} \qquad \in L$$

k als feste Zahl wählen, z.B. 10:

$$w \;=\; \underbrace{(\,(\,(\,...\,(}_{l+10\ mal} \qquad a \qquad \underbrace{,\,b\,)\,,b)\,...\,)}_{l+10\ mal} \qquad \in L$$

dann gilt:

∀ uvz **mit** w = uvz

u = $(^k$

v = $(^m$

z = $(^{[l+10-k-m]}$ a ,b) , b) ... , b)

dann wählenn wir:

w' = uv^0z = uz ∉ L

Denn es gehen [k+l+10-k-m = 10+l+m] Klammern auf und nur [l+10] Klammern zu. Da m≥ ist [10+l+m > 10+l]. D.h. w' ist nicht in 'L' und 'L' damit nicht regulär.