Konfluenz und Terminierung

Polynomordnung ⇒ System stark normalisierend

$(x+y)[2/x, 3/y] = 2+3$

1. Domäne $A = \mathbb{N}_{\geq n}, n \geq 1$ wählen
2. Für jedes Funktionssymbol ein stark monotones Polynom bzw. Konstante
3. Zeige dass für alle Produktionsregeln gilt

$$\text{links} \underset{A}{\succ} \text{rechts}$$

4. ggf. anpassen: $(x+c), (x \cdot c), x^2$
5. Das System ist stark normalisierend mit $\mathcal{A} = \{A, p_f, p_g, ...\}$auf der Domäne $A = \mathbb{N} \setminus \{...\}$

Kritische Paare

- Umbennen
- Für alle Regeln $l \to r$
  - $l_1 = l, C(\cdot) = (\cdot)$
  - Für alle sonstigen Regeln $l' \to r'$
    - $l_2 = l'$
    - `if` $\sigma = \text{mgu}(C(l_1), l_2)$
      σl_2 erzeugt ein kritisches paar mit r und r'
- ggf. $l_1 = l, C(\cdot) = ...$ (hier ist l_1 nochmal nicht mehr trivial)
  - wiederhole obigen Ablauf

Critical Pair Lemma: TES lokal konfluent, wenn alle kritischen Paare zusammenführbar sind.
Newtons Lemma:TES konfluent, wenn es lokal konfluent und stark normalisierend

Strukturelle Induktion

1. Zeige Aussage für das Nil-Element
2. Definiere ein X = Y + ... mit Annahme, Aussage gelte für Y
3. Zeige mit der Annahme dass Aussage für X gilt

Folds

fold c g Nil = c
fold c g (type t s) = g(fold c g t) (fold c g s)
fold n f g (typef x xs) = f x (fold c f g xs)
fold n f g (typeg x xs) = g x (fold c f g xs)

Fold-Funktion foldH von HList a b an.

- foldH: c $\to (a \to c \to c) \to (b \to c \to c) \to$ Hlist a b $\to c$
  foldH: n ca cb HNil = n
  foldH: n ca cb (ConsA x zs) = ca x (foldH n ca cb zs)
  foldH: n ca cb (ConsB y zs) = cb y (foldH n ca b zs)

mapA und mapB unter Verwendung von foldH an.

- mapA: $(a \to c) \to$ HList $ab \to$ Hlist c b
  mapA $f =$ foldH HNil $(\lambda x.\text{ConsA } (fx))$ Cons B
- mapB : $(b \to c) \to$ HList $ab \to$ Hlist a c
  mapB $g =$ foldH HNil Cons $A(\lambda y.\text{ConsB } (gy))$

Koinduktion

Eine Relation $R \subseteq (\text{Stream} \times \text{Stream})$ ist eine Bisimulation wenn für $(x, y) \in R$ gilt:

- *hd x = hd y*
- *(tl x) R (tl y)*

Zeige dass für alle s : StreamType gilt: *l s = r s*:

1. $R :=$*{ (l s, r s) |* $s \in$ StreamType *}*, zeige, dass R eine Bsm. ist:
2. Es muss gezeigt werden, dass *hd (l s) = ... = hd (r s)*
   und tl (l s) = ... = R ... s = tl (r s)

System F

Lambda-Kalkül linksassozziativ: a b c = (a b) c
Typen rechtsassoziativ $a \to b \to c = a \to (b \to c)$
$f : a \to b \to c \mid g : b \to c$

(Ax) $\dfrac{}{\Gamma \vdash x : \alpha}$	Es muss gelten: $(x : \alpha) \in \Gamma$
$(\to_i) \dfrac{\Gamma \cup \{(x : \alpha)\} \vdash s : \beta}{\Gamma \vdash \lambda x.s : \alpha \to \beta}$	Abstraktion entfernen und Typen zum Kontext hinzufügen
$\to_e \dfrac{\Gamma \vdash s : \alpha \to \beta \quad \Gamma \vdash t : \alpha}{\Gamma \vdash st : \beta}$	rechteste Applikation trennen. Rechter Typ = erster linker Typ
$(\forall_i) \dfrac{\Gamma \vdash s : \alpha}{\Gamma \vdash s : \forall a.\alpha}$	Es muss gelten: $\alpha \notin \text{FV}(\Gamma)$
$\forall_e \dfrac{\Gamma \vdash s : \forall a.\alpha}{\Gamma \vdash s : (\alpha[\beta \setminus \alpha])}$	Definition unten, mit ∀ definierter Typ oben

SystemF Beweise
Von unten nach Oben.

1. Unten: $\Gamma \vdash$ *funktion*: $T \to y \to p$ schreiben, dann f. in ihre Definition auflösen
2. ganz am Anfang mit $\forall_i$ ein ∀ entfernen
3. λ's solange mit $\to_i$ entfernen, und ggf. Typen mit $\forall_i$ lösen
4. rechteste Funktion vom Rest mit $\to_e$ trennen
5. Eine rechteste Menge an Funktionen mit (fgx) als eine behandeln

Äquivalenzen

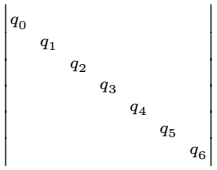
- α: Umbennung von Varaiblen z.B. $\lambda x.x \to_\alpha \lambda y.y (y \notin \text{FV}(t))$
- β: Argument anwenden z.B. $(\lambda xy.yx)a \to_\beta \lambda y.ya$
- σ: Ausdruck durch Defintion ersetzen, z.B. true $\to_\sigma \lambda xy.x$
- μ: Streichen ungenutzer Argumente, z.B. $\lambda x.y.x \to_\mu y$

Church Kodierungen

- $\mathbb{N}$:= *forall a . (a→a)→a→a*
- *zero:* $\mathbb{N}$ | *zero:* $\lambda f x .x$
- *suc :* $\mathbb{N} \to \mathbb{N}$ | *suc =* $\lambda n f x .f (n f x)$
- *fold:* $\forall$ *a. (a→a) -> a →* ($\mathbb{N} \to$ *a)* | *fold =* $\lambda f x n . n f x$
- *add:* $\mathbb{N} \to \mathbb{N} \to \mathbb{N}$ | *add =* $\lambda n .$ *fold suc n*
- *add: n m = fold suc n m*
- *pair :* $\forall$ *a b.* $a \to b \to$ *(a times b)* | *pair =* $\lambda x y f. f x y$
- *fst :* $\forall$ *a b. (a times b)* $\to a$ | *fst =* $\lambda p. p (\lambda x y .x)$
- *snd :* $\forall$ *a b. (a times b)* $\to b$ | *snd =* $\lambda p . p (\lambda x y. y)$
- *rev:* $\forall$ *a.* $\mathbb{L}$ *a* $\to \mathbb{L}$ *a* | *rev =* λ *l.l nil snoc*
- $(a + b) := \forall r.(a \to r) \to (b \to r) \to r$
  inl : $\forall ab.a \to (a+b) \to_a \text{inl} = \lambda \to xfg.fx$
  inr : $\forall ab.b \to (a+b) \to_b \text{inr} = yfg.gy$
- case : $\forall abs.(a \to s) \to (b \to s) \to (a+b) \to s \mid \text{case} = \lambda fgs.sfg$

Minimierung endlicher Automaten

1. Nicht-erreichbare Zustände ignorieren
2. Dreieckige Tabelle mit $q_0 - q_n$ in der Diagonalen aufstellen und nur rechts oben arbeiten
3. Alle Paare mit **nur einem** Endzustand mit 0 markieren



4. Alle Paare, die für dasselbe Zeichen in ein mit n markiertes Paar führen mit $n+1$ markieren (sollten Paare zu mehreren markierten Paaren führen, dann **Min.** wählen) 5. Unmarkierte Zustandspaare zusammenfassen $q_x, q_y, q_z \Longrightarrow q_{xyz}$

Beispiele:

- $La \coloneqq \forall r.r \to (a \to r \to r) \to r$
- nil : $\forall a.La \mid \text{nil} = \lambda rz.z$
- cons : $\forall a.a \to La \to La \mid \text{cons} = \lambda x\ \text{xs}.\lambda rf.fx(\text{xs}\ rf)$
- app : $\forall a.La \to La \to La \mid \text{app} = \lambda\ \text{xs ys}.\lambda rf.\ \text{xs}\ (\text{ys}\ rf)f$
- map : $\forall ab.(a \to b) \to La \to Lb \mid \text{map} = \lambda f\ \text{xs}.\lambda r\ \text{cons.}\ \text{xs}\ r(\lambda x\ \text{acc.}\ \text{cons}\ (fx)\ \text{acc})$
- rev: $\forall a.\mathbb{L}a \to \mathbb{L}a \mid \text{rev} = \lambda l.l\ \text{nil snoc}$
- length : $\forall a.La \to \text{Nat} \mid \text{length} = \lambda\ \text{xs.}\ \text{xs zero}\ (\lambda_n.\ \text{succ}\ n)$
- fold : $\forall ab.(a \to b \to b) \to b \to La \to b \mid \text{fold} = \lambda fz\ \text{xs.}\ \text{xs}\ zf$
- filter : $\forall a.(a \to \text{Bool}) \to La \to La \mid \text{filter} = \lambda p\ \text{xs}.\lambda r\ \text{cons.}\ \text{xs}\ r(\lambda x\ \text{acc.}\ \text{if}\ (px)\ \text{then cons}\ x\ \text{acc else acc})$
- head : $\forall a.La \to \text{Option}\ a \mid \text{head} = \lambda\ \text{xs.}\ \text{xs none}\ (\lambda x.\ \text{some}\ x)$
- empty : $\forall a.Ta \mid \text{empty} = \lambda r\ \text{node}.r$
- node : $\forall a.a \to Ta \to Ta \to Ta \mid \text{node} = \lambda x\ \text{left right}.\lambda rf.fx(\text{left}\ rf)(\text{right}\ rf)$
- preorder : $\forall a.Ta \to La \mid \text{preorder} = \lambda t.t\ \text{nil}\ (\lambda x\ \text{l1 l2.}\ \text{cons}\ x(\text{app l1 l2}))$
- inorder : $\forall a.Ta \to La \mid \text{inorder} = \lambda t.t\ \text{nil}\ (\lambda x\ \text{l1 l2.}\ \text{app l1}\ (\text{cons}\ x\ \text{l2}))$
- postorder : $\forall a.Ta \to La \mid \text{postorder} = \lambda t.t\ \text{nil}\ (\lambda x\ \text{l1 l2.}\ \text{app}\ (\text{app l1 l2})(\text{cons}\ x\ \text{nil}))$
- height : $\forall a.Ta \to \text{Nat} \mid \text{height} = \lambda t.\ \text{foldTree}\ (\lambda_l r.\succ(\max\ lr))\ \text{zero}\ t$
- size : forall $a.Ta \to \text{Nat} \mid \text{size} = \lambda t.\ \text{foldTree}\ (\lambda_l r.\succ(\text{add}\ lr))\ \text{zero}\ t$
- contains : $\forall a.(a \to \text{Bool}) \to Ta \to \text{Bool} \mid \text{contains} = \lambda pt.\ \text{foldTree}\ (\lambda xlr.\vee(px)(\vee lr))\ \text{false}\ t$
- some: $\forall a.a \to \text{Maybe}\ a \mid \lambda x.\lambda fn.fx$
- none: $\forall a.\ \text{Maybe a}\ \lambda fx.x$
- *foldNT: (Nat ⟶ b) ⟶ (b ⟶ b ⟶ b) ⟶ NatTree ⟶ b | foldNT f g (leaf x) = f x*
  foldNT f g (branch l r) = g (foldNT f g l) (foldNT f g r)

rev: $\forall a.\mathbb{L}a \to \mathbb{L}a \mid \text{rev} = \lambda l.l\ \text{nil snoc}$,

$$\dfrac{\dfrac{\dfrac{}{\Gamma,(l:\mathbb{L}a)\vdash l:\mathbb{L}a}\,Ax}{\Gamma,(l:\mathbb{L}a)\vdash l:\mathbb{L}a\to(a\to\mathbb{L}a\to\mathbb{L}a)\to\mathbb{L}a}\,\forall_e \quad \dfrac{\dfrac{}{\Gamma,(l:\mathbb{L}a)\vdash \text{nil}:\forall a.\mathbb{L}a}\,Ax}{\Gamma,(l:\mathbb{L}a)\vdash \text{nil}:\mathbb{L}a}\,\forall e}{\Gamma,(l:\mathbb{L}a)\vdash l\ \text{nil}\ (a\to\mathbb{L}a\to\mathbb{L}a)\to\mathbb{L}a}\,\to_e \quad \dfrac{\dfrac{}{\Gamma,(l:\mathbb{L}a)\vdash \text{snoc}:\forall a.a\to\mathbb{L}a\to\mathbb{L}a}\,Ax}{\Gamma,(l:\mathbb{L}a)\vdash \text{snoc}:a\to\mathbb{L}a\to\mathbb{L}a}\,\forall e}{\dfrac{\dfrac{\Gamma,(l:\mathbb{L}a)\vdash l\ \text{nil snoc}:\mathbb{L}a}{\Gamma\vdash\lambda l.\,l\ \text{nil snoc}:\mathbb{L}a\to\mathbb{L}a}\,\to_i}{\Gamma\vdash \text{rev}:\forall a.\mathbb{L}a\to\mathbb{L}a}\,\forall i}\,\to_e$$

$\Gamma \coloneqq (x:a),(l:\mathbb{L}a),(u:r),(f:a\to r\to r)$

$$\dfrac{\dfrac{\dfrac{}{\Gamma\vdash l:\mathbb{L}a}\,Ax}{\Gamma\vdash l:r\to(a\to r\to r)\to r}\,\forall_e \quad \dfrac{\dfrac{}{\Gamma\vdash f:a\to r\to r}\,Ax \quad \dfrac{}{\Gamma\vdash x:a}\,Ax}{\Gamma\vdash fx:r\to r}\,\to e \quad \dfrac{}{\Gamma\vdash u:r}\,Ax}{\dfrac{\Gamma\vdash l(fxu):(a\to r\to r)\to r}{\Gamma\vdash l(fxu)f:r}\,\to_e \quad \dfrac{}{\Gamma\vdash f:a\to r\to r}\,Ax}\,\to_e$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{(x:a),(l:\mathbb{L}\,a),(u:r),(f:a\to r\to r)\vdash l\ (f\ x\ u)\ f:r}{(x:a),(l:\mathbb{L}\,a),(u:r)\vdash\lambda\,f.\,l\ (f\ x\ u)\ f:(a\to r\to r)\to r}\,\to_i}{(x:a),(l:\mathbb{L}\,a)\vdash\lambda\,u\,f.\,l(f\ x\ u)f:r\to(a\to r\to r)\to r}\,\to_i}{(x:a),(l:\mathbb{L}\,a)\vdash\lambda\,u\,f.\,l\ (f\ x\ u)f:\mathbb{L}\,a}\,\forall_i}{(x:a)\vdash\lambda\,l.\,\lambda\,u\,f.\,l\ (f\ x\ u)\ f:\mathbb{L}\,a\to\mathbb{L}\,a}\,\to_i}{\vdash\lambda\,x\,l.\,\lambda\,u\,f.\,l\ (f\ x\ u)\ f:a\to\mathbb{L}\,a\to\mathbb{L}\,a}\,\to_i}{\vdash\text{snoc}:\forall\,a.\,a\to\mathbb{L}\,a\to\mathbb{L}\,a}\,\forall_i$$

Definieren sie eine Funktion left and right:

```
left: BiStream a b -> Stream a
hd ( left as ) = lhd as
tl ( left as ) = left ( btl as )

right: BiStream a b -> Stream a
hd ( right as ) = rhd as
tl ( right as ) = right ( btl as )
```

Schreiben Sie eine korekursive Funktion f, so dass f 1 den Baum wie folgt befüllt: [1, [2, [4, 5], [3, [6, 7]]]

```
node (f x) = x
left (f x) = f ( 2*x )
right (f x) = f ( 2*x + 1)
```

Definieren Sie korekursiv die Funktion fanout: Stream a ⟶ Tree a, die aus einem Stream einen Baum macht

```
node(fanout s) = hd(s)
left(fanout s) = fanout(tl s)
right(fanout s) = fanout(tl s)
```

Stream nat. Zahlen

```
nat: Stream Nat
  hd nat = Zero
  tl nat = natr (Suc zero)

natr: Nat -> Stream Nat
  hd(natr n) = n
  tl (natr n) = natr (Suc n)
```

Def. xor

```
xor: Bitstream -> Bitstream -> Bitstream
cur(xor x y) = (cur x) o.plus (cur y)
next(xor x y) = xor (next x) (next y)
```

$$R \coloneqq \{(\text{map tl}\ (\text{transpose}\ s), \text{transpose}\ (\text{tl}\ s)) \mid s \in \mathsf{Stream}\ a\}$$

und zeigen, dass R eine Bisimulation ist. Sei also $(\text{map tl}\ (\text{transpose}\ s), \text{transpose}\ (\text{tl}\ s)) \in R$:

- $\text{hd}\ (\text{map tl}\ (\text{transpose}\ s))$
  $= \text{tl}\ (\text{hd}\ (\text{transpose}\ s))$
  $= \text{tl}\ (\text{map hd}\ s)$
  $= \text{map hd}\ (\text{tl}\ s)$
  $= \text{hd}\ (\text{transpose}\ (\text{tl}\ s))$
- $\text{tl}\ (\text{map tl}\ (\text{transpose}\ s))$
  $= \text{map tl}\ (\text{tl}\ (\text{transpose}\ s))$
  $= \text{map tl}\ (\text{transpose}\ (\text{map tl}\ s))$
  $R\ \text{transpose}(\text{tl}\ (\text{map tl}\ s))$
  $= \text{transpose}\ (\text{map tl}\ (\text{tl}\ s))$
  $= \text{tl}\ (\text{transpose}\ (\text{tl}\ s))$