Schemata

10.10.2023 - ThProg - SoSe 2023

1. TES

Kritische Paare

Konfluent: stark normalisierend und lokal konfluent
Lokal konfluent: alle kritischen Paare sind zusammenführbar

Definition 2.57. Seien $l_1 \to_0 r_1$ und $l_2 \to_0 r_2$ zwei Umformungsregeln des Termersetzungssystems sowie $FV(l_1) \cap FV(l_2) = \emptyset$ (ggf. nach Umbenennung). Sei $l_1 = C(t)$, wobei t ein nichttrivialer Term ist (d.h. t ist nicht nur eine Variable), so dass t und l_2 unifizierbar sind. Sei $\sigma = \mathsf{mgu}(t, l_2)$. Dann heißt $(r_1\sigma, C(r_2)\sigma)$ ein *kritisches Paar*. Ein solches kritisches Paar ist *trivial*, wenn die beiden Umformungsregeln $l_1 \to_0 r_1$ und $l_2 \to_0 r_2$ bis auf Umbenennung gleich sind und $C(\cdot) = (\cdot)$.

Step by Step Anleitung:

- alle linken Seiten miteinander kombinieren
    - falls l1 und l2 keine gemeinsamen Funktionssymbole haben keine krit. Paare
    - falls l1 kein Symbol doppelt hat dann keine krit. Paare
- Alle Kontexte probieren
    - Erst C(.) = (.) mit t = l1
    - Schritt fuer Schritt Funktionen aus l1 in den Kontext reinziehen. t ist der Rest (.)
    - falls t = x1 trivial => Paar ist egal
    - falls C(.) = (.) und die Regeln bis auf Umbenennung gleich trivial => Paar ist egal
- $(r_1\sigma, C(r_2)\sigma)$ Wenn man t mit mgu in l2 umwandeln kann oder umgekehrt krit Paar gefunden
- Es hat die Form
- Wenn alle Paare zusammenfuehrbar sind (man kommt durch Regelanwendungen auf selbe Form) dann ist das System konfluent

Bei Regelanwendungen zur Zusammenführbarkeit glaube ich immer Regel, Kontext und die Umbenennung (von Regel zu eigenem Fall) angeben
(C(.) = (.) darf man weglassen)

Gegenbeispiel: Endlosschleife oder endlose Funktionsanwendung

Starke Normalisierung = Terminierung: Es gibt mind. Eine Regelkette die zu einer Normalform führt

Schwache Normalisierung: Alle Regelketten führen zu einer Normalform

Schwache Normalisierung + Krit. Paare zusammenführbar = Lokal Konfluent

Starke Normalisierung + Krit. Paare zusammenführbar = Lokal Konfluent -> Newman‘s Lemma -> Konfluent

2. SYSTEM F

Lambda Kalkül (links vom Doppelpunkt) ist linksassozziativ: a b c = (a b) c
Typen (rechts vom Doppelpunkt) rechtsassoziativ: a → b → c = a → (b → c)

let a = b c in a x b a , ist einfach (b c) x b (b c) also nur a durch b c ersetzen

$$(Ax)\ \frac{}{\Gamma \vdash x : \alpha}\ x : \alpha \in \Gamma$$

$$(\to_e)\ \frac{\Gamma \vdash t : \alpha \to \beta \qquad \Gamma \vdash s : \alpha}{\Gamma \vdash ts : \beta}$$

$$(\to_i)\ \frac{\Gamma[x \mapsto \alpha] \vdash t : \beta}{\Gamma \vdash \lambda x.\, t : \alpha \to \beta}$$

$$(\forall_i)\ \frac{\Gamma \ \vdash \ s : \alpha \qquad a \notin FV(\Gamma)}{\Gamma \ \vdash \ s : \forall a.\alpha}$$

- $$(\forall_e)\ \frac{\Gamma \ \vdash \ s : \forall a.\alpha}{\Gamma \ \vdash \ s : (\alpha[\beta/a])}$$

2.1 PT-ALGORITHMUS

1. $PT(\Gamma \vdash x : \alpha) = \{\alpha \doteq \beta \mid x : \beta \in \Gamma\}$
2. $PT(\Gamma \vdash \lambda x.\, t : \alpha) = PT((\Gamma[x \mapsto a]) \vdash t : b) \cup \{a \to b \doteq \alpha\}$, mit a, b frisch.
3. $PT(\Gamma \vdash ts : \alpha) = PT(\Gamma \vdash t : a \to \alpha) \cup PT(\Gamma \vdash s : a)$, mit a frisch.

2.2 ÄQUIVALENZEN

α-Äquivalenz: bis auf Umbenennung gebundener Variablen gleich: λx. xy =α λz. Zy

β-Reduktion: ein einfügen bei λx. t also bspw. (λx. xx) a —>β aa

δ-Reduktion: Definition einsetzen: flip const twice ->δ (λfxy.fyx) const twice , mit flip = λfxy.fyx

Applikativ:
Das linkeste innerste zuerst

Normal:
Das linkeste äußerste zuerst

Definition 3.13. Die *applikative (auch: leftmost-innermost) Reduktion* $\to_a$ ist induktiv definiert durch:

- $(\lambda x.\, t)s \to_a t[s/x]$, wenn t und s normal sind.
- $\lambda x.\, t \to_a \lambda x.\, t'$, wenn $t \to_a t'$.
- $ts \to_a t's$, wenn $t \to_a t'$.
- $ts \to_a ts'$, wenn $s \to_a s'$ und t normal ist.

Definition 3.14. Die *normale (auch: leftmost-outermost) Reduktion* $\to_n$ ist definiert durch

- $(\lambda x.\, t)s \to_n t[s/x]$.
- $\lambda x.\, t \to_n \lambda x.\, t'$, wenn $t \to_n t'$.
- $ts \to_n t's$, wenn $t \to_n t'$ und t keine λ-Abstraktion ist.
- $ts \to_n ts'$, wenn $s \to_n s'$ und t normal und keine λ-Abstraktion ist.

3. Strukturelle Induktion

Weiterhin linksassozziativ: a b c = (a b) c

```
data List a where
    Nil  : () → List a
    Cons : a → List a → List a
```

```
data Tree a where
    Leaf  : a → Tree a
    Inner : a → Tree a → Tree a → Tree a
```

Manchmal auch: data Nat = Z | S Nat

In jedem = Schritt sagen was benutzt wurde

Induktion über xs **(hinschreiben)**:

IA: (xs = Nil): links = … = rechts

IV: (*) mit zs drin

IS: (xs = Cons z zs): links = … = rechts mit Hilfe von data Nat = Z | S Natn IV

3.1 Fold-Funktion

$$foldL\ :\ c\ \to (a \to c \to c) \to \mathbf{List}\ a \to c$$
$$foldL\ \ n\ f\ \ Nil\ \ = n$$
$$foldL\ \ n\ f\ \ (Cons\ x\ xs)\ \ = f\ x\ \ (foldL\ \ n\ f\ \ xs)$$

4. Koinduktion

Head = Erstes Element, Tail = Rest vom Stream ohne das erste Element

$R = \{(linke\ Seite, rechte\ Seite) \mid n, m \in Nat\}$ R ist Bisimulation, wenn gilt $\forall a,b.\ aRb \to hd\ a = hd\ b \wedge (tl\ a)\ R\ (tl\ b)$

Daraus folgt, wenn R eine Bisimulation ist $sRt \implies s = t$

Dann beide Seiten in die einsetzen und entsprechend der Definition von der Bisimulation umformen.

Evtl. Noch unter Bisimulation R‘, falls bei tl a R tl b stuck: $R' = \{(tl\ a,\ tl\ b \mid n, m \in Nat)\} \cup R$

Definition 4.39. Eine Relation $R \subseteq A^\omega \times A^\omega$ heißt *Bisimulation*, wenn für alle $(s,t) \in R$ gilt:

- $hd\ s = hd\ t$
- $(tl\ s)\ R\ (tl\ t)$

Kodatentyp der eine Liste erzeugt die abwechseln a,b ist (a,b,a,b,…)

alt: Stream Char

```
  hd alt = ‚a‘
  tl alt = helper
    where helper: Stream Char
      hd helper = ‚b‘
      tl helper = alt
```

alt Bool -> Stream

hd (alt state) = if state then ‚a‘ else ‚b‘

tl (alt state) = alt (not state)

5. AUTOMATENMINIMIERUNG

Zeichne diese Tabelle und lasse unerreichbare Zustände weg

1. Wenn qi und qj sich unterscheiden (Endzustand bzw. normaler Zustand) trage 0 ein sonst leer
2. Schaue für jedes freie Feld und jede Eingabe die Zustände die rauskommen an
   (bspw. Feld af dann nachschauen δ(a, 0) = e und δ(f, 0) = g)
   Dann schau an Stelle eg nach und falls dort eine Zahl steht, dann schreibe die Anzahl der Durchläufe (am Anfang 1) in af
   (hier werden nur Zahlen beachtet die vor dem aktuellen Durchlauf drin waren also keine Zahl die man gerade erst eingetragen hat. Außerdem sind die Felder aa, bb, ... immer leer)
3. Wiederhole 3 bis sich nichts ändert
4. Lege Zustände mit leerem Feld zusammen (Feld ab leer => ab in einen Zustand)

Algorithmus erkennt unerreichbare Zustände nicht automatisch!!!

Beim Zeichnen nicht Start- und Endzustände vergessen!

6. REIHENFOLGE

1. Korekursion & Koinduktion
2. Strukturelle Induktion
3. System F / TES
4. Automatenminimierung
5. TES