Az egyszerűen típusozott lambda kalkulus kiértékeléses normalizálásának formális helyességbizonyítása

Szerző: Kovács András Témavezető: Kaposi Ambrus

Eötvös Loránd Tudományegyetem

2019

Eredmények

- Egyszerű típusos lambda kalkulus $\beta\eta$-normalizálása.

Eredmények

- Egyszerű típusos lambda kalkulus $\beta\eta$-normalizálása.
- A leírt algoritmus helyessége formálisan verifikált, Agda rendszerben.

Eredmények

- Egyszerű típusos lambda kalkulus $\beta\eta$-normalizálása.
- A leírt algoritmus helyessége formálisan verifikált, Agda rendszerben.
- Korábbi hasonló formalizálásoknál lényegesen tömörebb, hatékony algoritmusra és standard (tankönyvi) szintaxisra.

Egyszerű típusos lambda kalkulus (STLC)

Magasabbrendű függggvények minimális programozási nyelve.

Egyszerű típusos lambda kalkulus (STLC)

Magasabbrendű függggvények minimális programozási nyelve.

Informális példák kifejezésekre:

```
λ (x : ι). x
λ (f : ι → ι). λ (g : ι → ι). λ (x : ι). f (g x)
(λ (f : ι → ι). f) (λ (x : ι) → x)
```

Egyszerű típusos lambda kalkulus (STLC)

Magasabbrendű függggvények minimális programozási nyelve.

Informális példák kifejezésekre:

```
λ (x : ι). x
λ (f : ι → ι). λ (g : ι → ι). λ (x : ι). f (g x)
(λ (f : ι → ι). f) (λ (x : ι) → x)
```

Normálformák: nem végezhető el több függvényalkalmazás és η-kifejtés.

Egyszerű típusos lambda kalkulus (STLC)

Magasabbrendű függggvények minimális programozási nyelve.

Informális példák kifejezésekre:

```
λ (x : ι). x
λ (f : ι → ι). λ (g : ι → ι). λ (x : ι). f (g x)
(λ (f : ι → ι). f) (λ (x : ι) → x)
```

Normálformák: nem végezhető el több függvényalkalmazás és η-kifejtés.

Normalizálás jelentősége más területeken: logika, kategóriaelmélet, típusellenőrzés polimorf és függő típusos nyelvekhez.

Kiértékeléses normalizálás

Normalization-by-evaluation (NbE).

Kiértékeléses normalizálás

Normalization-by-evaluation (NbE).

Alapötlet: naiv helyettesítések helyett használjunk hatékony absztrakt gépet, a szintaxist kiértékeljük futásidejű objektumokra, majd ebből a normálformákat visszaolvassuk.

Kiértékeléses normalizálás

Normalization-by-evaluation (NbE).

Alapötlet: naiv helyettesítések helyett használjunk hatékony absztrakt gépet, a szintaxist kiértékeljük futásidejű objektumokra, majd ebből a normálformákat visszaolvassuk.

A zárt funkcionális programok standard kiértékelését (pl. GHC, OCaml) általánosítja nyílt (szabad változókat tartalmazó) programokra.

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

(1) Complete: a függvény kimenete $\beta\eta$-ekvivalens a bemenettel.

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

(1) Complete: a függvény kimenete $\beta\eta$-ekvivalens a bemenettel.
(2) Sound: a függvény $\beta\eta$-ekvivalens bemeneteket azonos kimenetre képez.

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

(1) Complete: a függvény kimenete $\beta\eta$-ekvivalens a bemenettel.
(2) Sound: a függvény $\beta\eta$-ekvivalens bemeneteket azonos kimenetre képez.
(3) Stable: a függvény nem csinál semmit, ha az input eleve normálforma.

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

(1) Complete: a függvény kimenete $\beta\eta$-ekvivalens a bemenettel.
(2) Sound: a függvény $\beta\eta$-ekvivalens bemeneteket azonos kimenetre képez.
(3) Stable: a függvény nem csinál semmit, ha az input eleve normálforma.

Helyesség fogalma

Három feltétel egy normálformákat visszaadó függvényre:

(1) Complete: a függvény kimenete $\beta\eta$-ekvivalens a bemenettel.
(2) Sound: a függvény $\beta\eta$-ekvivalens bemeneteket azonos kimenetre képez.
(3) Stable: a függvény nem csinál semmit, ha az input eleve normálforma.

A három feltételből következik, hogy minden kifejezésnek pontosan egy normálformája van.

Korábbi formalizálások hátrányai

- Big-step normalization (Altenkirch & Chapman [1]): nem standard szintaxis (explicit szubsztitúció), nem teljes formalizálás.
- Catarina Coquand [2]: explicit szubsztitúció, formalizálás legacy rendszerben (ALF).
- Hereditary substitution (Altenkirch & Keller [3]): nem hatékony algoritmus, bonyolult formalizálás.

Sorok száma (nagyjából)

Formalizáció	Sorok
NbE (Kovács)	600
Big-step (Romanenko [4])	1600
Hereditary substitution (Altenkirch & Keller [3])	1200
NbE (Catarina Coquand [2])	?

Szintaxis

Függvények (`A ⇒ B`) és egy üres alaptípus (`ι`).

De Bruijn indexek.

Kizárólag jól típusozott termekkel foglalkozunk.

A helyettesítés rekurzív függvény, a $\beta\eta$-konverzió pedig induktív reláció a szintaxison.

Formális szintaxis

```
data Ty : Set where
  ι   : Ty
  _⇒_ : Ty → Ty → Ty

data Con : Set where
  •   : Con
  _,_ : Con → Ty → Con

data _∈_ (A : Ty) : Con → Set where
  vz : A ∈ (Γ , A)
  vs : A ∈ Γ → A ∈ (Γ , B)

data Tm (Γ : Con) : Ty → Set where
  var : A ∈ Γ → Tm Γ A
  lam : Tm (Γ , A) B → Tm Γ (A ⇒ B)
  app : Tm Γ (A ⇒ B) → Tm Γ A → Tm Γ B
```

Formalizálás elvei

A helyettesítések és változó-átnevezések formalizálása vesződséges, könnyű benne elveszni.

Formalizálás elvei

A helyettesítések és változó-átnevezések formalizálása vesződséges, könnyű benne elveszni.

A részletekbe bonyolódás helyett kiválasztunk egy olyan absztrakciós szintet, ami:

Formalizálás elvei

A helyettesítések és változó-átnevezések formalizálása vesződséges, könnyű benne elveszni.

A részletekbe bonyolódás helyett kiválasztunk egy olyan absztrakciós szintet, ami:

(1) Elég absztrakt és világos ahhoz, hogy könnyű legyen áttekinteni.

Formalizálás elvei

A helyettesítések és változó-átnevezések formalizálása vesződséges, könnyű benne elveszni.

A részletekbe bonyolódás helyett kiválasztunk egy olyan absztrakciós szintet, ami:

(1) Elég absztrakt és világos ahhoz, hogy könnyű legyen áttekinteni.
(2) Viszont nem *túl absztrakt* olyan értelemben, hogy még egyszerűen leírható Agda-ban.

Lépések

(1) Megmutatjuk, hogy a szintaxis a $\beta\eta$-konverzióval és a párhuzamos helyetessítésekkel együtt sCwF (simply typed category with families) algebrai struktúrát alkot.

Lépések

(1) Megmutatjuk, hogy a szintaxis a $\beta\eta$-konverzióval és a párhuzamos helyetessítésekkel együtt sCwF (simply typed category with families) algebrai struktúrát alkot.
(2) Megadjuk az így kapott sCwF presheaf modelljét, ahol a bázis a típuskörnyezetek és a változó-átnevezések kategóriája. Ez megadja a kiértékelés algoritmusát.

Lépések

(1) Megmutatjuk, hogy a szintaxis a $\beta\eta$-konverzióval és a párhuzamos helyetessítésekkel együtt sCwF (simply typed category with families) algebrai struktúrát alkot.
(2) Megadjuk az így kapott sCwF presheaf modelljét, ahol a bázis a típuskörnyezetek és a változó-átnevezések kategóriája. Ez megadja a kiértékelés algoritmusát.
(3) Bevezetünk egy logikai relációt a szintaxis és a presheaf modell között, aminek az alaplemmája megadja a "completeness" tulajdonságot.

Lépések

(1) Megmutatjuk, hogy a szintaxis a $\beta\eta$-konverzióval és a párhuzamos helyetessítésekkel együtt sCwF (simply typed category with families) algebrai struktúrát alkot.
(2) Megadjuk az így kapott sCwF presheaf modelljét, ahol a bázis a típuskörnyezetek és a változó-átnevezések kategóriája. Ez megadja a kiértékelés algoritmusát.
(3) Bevezetünk egy logikai relációt a szintaxis és a presheaf modell között, aminek az alaplemmája megadja a “completeness” tulajdonságot.
(4) A “soundness” és “stability” egyszerű indukcióval bizonyítható a szintaxis fölött.

Tanulságok, összefüggések

Erre az egyszerű nyelvre is meglepően bonyolult a normalizálás helyességbizonyítása.

Tanulságok, összefüggések

Erre az egyszerű nyelvre is meglepően bonyolult a normalizálás helyességbizonyítása.

A bizonyítás elég absztrakt, de megéri.

Tanulságok, összefüggések

Erre az egyszerű nyelvre is meglepően bonyolult a normalizálás helyességbizonyítása.

A bizonyítás elég absztrakt, de megéri.

Szoros összefüggés logika és programozás között. Az STLC konstruktív propozícionális logika, az STLC algoritmusai megfelelnek bizonyos logikai tulajdonságok bizonyításának.

Tanulságok, összefüggések

Erre az egyszerű nyelvre is meglepően bonyolult a normalizálás helyességbizonyítása.

A bizonyítás elég absztrakt, de megéri.

Szoros összefüggés logika és programozás között. Az STLC konstruktív propozícionális logika, az STLC algoritmusai megfelelnek bizonyos logikai tulajdonságok bizonyításának.

szemantika	algoritmus	logika
standard	jól típusozott interpreter	konzisztencia
Kripke	normalizálás	teljesség
presheaf	normalizálás + helyesség	teljesség/függetlenség

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.
- Új alaptípusok bevezetése.

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.
- Új alaptípusok bevezetése.
  - Szorzat és összeg típusok, algebrai típusok.

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.
- Új alaptípusok bevezetése.
  - Szorzat és összeg típusok, algebrai típusok.
- Polimorf és függő típusú nyelvek normalizálása: a jelenlegi módszer sajnos *nem elég absztrakt* ehhez.

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.
- Új alaptípusok bevezetése.
  - Szorzat és összeg típusok, algebrai típusok.
- Polimorf és függő típusú nyelvek normalizálása: a jelenlegi módszer sajnos *nem elég absztrakt* ehhez.
  - A megfelelő absztrakciós szintet nem támogatják jelenlegi bizonyítórendszerek.

Kérdések, további munka

- További egyszerűsítés automatikus bizonyítással segítségével. Létező könyvtárak Agda-ban és máshol.
- Új alaptípusok bevezetése.
  - Szorzat és összeg típusok, algebrai típusok.
- Polimorf és függő típusú nyelvek normalizálása: a jelenlegi módszer sajnos *nem elég absztrakt* ehhez.
  - A megfelelő absztrakciós szintet nem támogatják jelenlegi bizonyítórendszerek.
  - Jelenlegi kutatási téma ezeknek az absztrakciós eszközöknek az elmélete és implementációja.

[1] T. Altenkirch and J. Chapman, "Big-step normalisation," *Journal of Functional Programming*, vol. 19, nos. 3-4, pp. 311–333, 2009.

[2] C. Coquand, "A formalised proof of the soundness and completeness of a simply typed lambda-calculus with explicit substitutions," *Higher-Order and Symbolic Computation*, vol. 15, no. 1, pp. 57–90, 2002.

[3] C. Keller and T. Altenkirch, "Normalization by hereditary substitutions," *proceedings of Mathematical Structured Functional Programming*, 2010.

[4] S. Romanenko, "Big-step normalization," 2016. [Online]. Available: https://github.com/sergei-romanenko/chapman-big-step-normalization. [Accessed: 08-May-2017].