A Machine-Checked Correctness Proof of Normalization by Evaluation for Simply Typed Lambda Calculus

Szerző: Kovács András   Témavezető: Kaposi Ambrus

Eötvös Loránd Tudományegyetem

2017 tavaszi informatikai kari TDK

Célok

- Simply Typed Lambda Calculus (STLC), $\beta\eta$-normalizálás helyessége
- Formális, teljes, "csalás" nélkül
- Szokásos szintaxis (nincs explicit szubsztitúció)
- Normalizálás: minél egyszerűbb önálló implementáció
  - (szintaxis + normalizálás: 100 sor Agda)
- Helyesség a lehető legegyszerűbben (500-600 sor Agda)
- Normalization by evaluation (NbE): hatékony, viszonylag könnyű bizonyítás, könnyű η-normalizálás (vö. erős $\beta\eta$-redukció + Church-Rosser).

Meglévő formalizálások hátrányai

- Big-step normalization (Altenkirch & Chapman): explicit szubsztitúció, nem strukturálisan rekurzív, némi csalás.
- NbE (C. Coquand): explicit szubsztitúció, formalizálás legacy rendszerben (ALF).
- Hereditary substitution (Altenkirch & Keller): rettentően bonyolult, lassú algoritmus.

Sorok száma (nagyjából)

- Saját: ~700 (alap verzió), ~600 (tömör verzió)
- Big-Step (Altenkirch & Chapman [1]): ~830
- Big-Step (Romanenko [2]): ~1600
- Hereditary Substitution (Altenkirch & Keller [3]): ~1200
- NbE (Catarina Coquand [4]): ?

Szintaxis

- Függvények + egy üres alaptípus.
- De Bruijn indexek.
- Intrinzikus: kizárólag jól típusozott termekkel foglalkozunk.

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
```

Alapötlet operációs szemszögből

- A meta-nyelvben elve implementálva vannak a magasabbrendű függvények: használjuk fel ezt.
- Könnyű jól típusozott interpretert írni STLC-re
  - De ez "gyenge" kiértékelés: lambdák testét nem redukálja.
- Megoldás: interpreter "extra struktúrával".
- Szükség van friss változókra, hogy lambdák alá tudjon menni a normalizálás.
- A függvények interpretációi extra paraméterként kapnak "forrást" friss változókhoz.
- Minél több struktúrát adunk az interpreterhez, annál több lehetőségünk van.
  - Standard interpreter -> gyenge kiértékelés
  - Kripke -> normalizálás
  - Presheaf -> normalizálás helyességbizonyítással

Standard ("Set") interpreter

```
⟦_⟧ : Ty → Set
⟦ ι ⟧     = ⊥
⟦ A ⇒ B ⟧ = ⟦ A ⟧ → ⟦ B ⟧

⟦_⟧ : Con → Set
⟦ • ⟧     = ⊤
⟦ Γ , A ⟧ = ⟦ Γ ⟧ × ⟦ A ⟧

⟦_⟧ : ∀ {Γ A} → A ∈ Γ → Conˢ Γ → Tyˢ A
⟦ vz   ⟧ (⟦Γ⟧ , ⟦t⟧) = ⟦t⟧
⟦ vs v ⟧ (⟦Γ⟧ , _   ) = ⟦ v ⟧ ⟦Γ⟧

⟦_⟧ : ∀ {Γ A} → Tm Γ A → Conˢ Γ → Tyˢ A
⟦ var v   ⟧ ⟦Γ⟧ = ⟦ v ⟧ ⟦Γ⟧
⟦ lam t   ⟧ ⟦Γ⟧ = λ ⟦a⟧ → ⟦ t ⟧ (⟦Γ⟧ , ⟦a⟧)
⟦ app f a ⟧ ⟦Γ⟧ = ⟦ f ⟧ ⟦Γ⟧ (⟦ a ⟧ ⟦Γ⟧)
```

Kripke interpretáció (csak típusokra itt)

```
⟦_⟧ : Ty → Con → Set
⟦ ι     ⟧ Γ = Nf Γ ι
⟦ A ⇒ B ⟧ Γ = ∀ {Δ} → OPE Δ Γ → ⟦ A ⟧ Δ → ⟦ B ⟧ Δ
```

- `OPE`: order-preserving context embedding.
- A függvény interpretációja bármilyen *kiterjesztett* környezetre van általánosítva.
- Friss változókat úgy kapunk, hogy az `OPE`-vel kiterjesztjük a környezetet.

Presheaf interpretáció

- Helyességbizonyításhoz szükség van arra is, hogy a kiértékelés és az `OPE`-vel való gyengítés kommutálnak ("természetesség").
- A Kripke függvények között vannak "természetellenesek", pl. olyan függvény, ami más termet ad vissza az input Δ környezet hosszától függően.
- Presheaf: megszorítja a szemantikus függvényeket a természetesekre.

Presheaf interpretáció (csak típusokon itt)

```
mutual
  ⟦_⟧ : Ty → Con → Set
  ⟦ ι      ⟧ Γ = Nf Γ ι
  ⟦ A ⇒ B ⟧ Γ =
    Σ (∀ {Δ} → OPE Δ Γ → ⟦ A ⟧ Δ → ⟦ B ⟧ Δ)
    λ ⟦f⟧ →
        ∀ {Δ Σ}(σ : OPE Δ Γ)(δ : OPE Σ Δ) ⟦a⟧
      → ⟦f⟧ (σ ∘ δ) (embed δ ⟦a⟧) ≡ embed δ (⟦f⟧ σ ⟦a⟧)

  embed : ∀ {A Γ Δ} → OPE Δ Γ → ⟦ A ⟧ Γ → ⟦ A ⟧ Δ
  -- implementáció kihagyva
```

Direkt/indirekt presheaf

- Két lehetőség:
  1. Rögtön presheaf modellel normalizálunk. Ez elegánsabb és tömörebb, de egyszerre kell az algoritmust és a helyesség egy részét definiálni.
  2. Kripke modellel normalizálunk, aztán külön finomítjuk a kiértékelés értékkészletét presheaf modellre. Ezzel az algoritmus definíciója szép és egyszerű, de többet kell bizonyítani.
- Mindkettő formalizálva.

Kérdések

- Más tételbizonyító rendszeren lehet-e, egyszerűbb-e?
  - Coq, Idris OK. Nem egyszeűbb lényegesen.
- Új alaptípusok bevezetése.
  - Szigorú pozitív ADT-k könnyen bevezethetők.
  - Polimorfizmus és függőség nagyságrendileg nehezebb.
- Polimorf típusok normalizálása: mennyire nehéz, irodalom mit szól?
  - Nehéz, jelenlegi kutatás témája.
  - System F NbE algoritmus (saját).
  - Függő típusok: informálisan sok mindenre van NbE, formálisan csak Altenkirch & Kaposi.
  - Feladat: formálisat közelíteni az informálishoz.

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

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

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

[4] 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.