There is no finite model of the SK combinator calculus

Zhenyun Yin

Supervisor : Ambrus Kaposi

Eötvös Loránd University

May 29, 2024

Overview

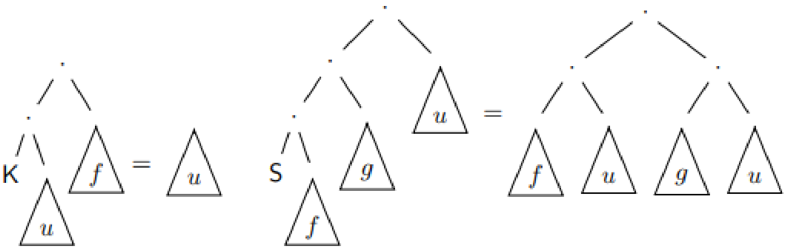
- **Goal**: To prove there is no finite non-trivial model in SK combinator calculus using machine verification.
- **Proof assistant**: Agda

SK combinator calculus

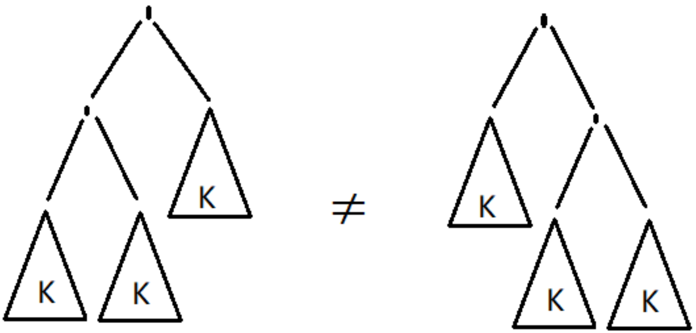
- **In CL**:
  - $\mathsf{t} ::= \mathsf{K} \mid \mathsf{S} \mid (\mathsf{t}\ \mathsf{t})$
  - $(\mathsf{K}u)f = u$
  - $((\mathsf{S}f)g)u = (fu)(gu)$
- Turing-complete language without variables.

Combinator calculus syntax

$\mathsf{K}uf = u$ $\mathsf{S}fgu = fu(gu)$



$(KK)K \neq K(KK)$



Programming with combinators

Identity Function (I)

- $\mathsf{I} = \mathsf{SKK}$
- $\mathsf{I}x = \mathsf{SKK}x$
  $= \mathsf{K}x(\mathsf{K}x)$
  $= x$

Function Composition (B)

- $\mathsf{B} = \mathsf{S(KS)K}$
- $\mathsf{B}fgu = \mathsf{S(KS)K}fgu$
  $= \mathsf{KS}f(\mathsf{K}f)gu$
  $= \mathsf{S}(\mathsf{K}f)gu$
  $= \mathsf{K}fu(gu)$
  $= f(gu)$

Connection to lambda calculus

1. Introduced by Schönfinkel 1924
2. Lambda calculus: Church 1928
3. They are equivalent (Rosser 1935)
4. Turing machine : Turing 1936
5. Combinatory Logic: Pure, Applied and Typed: Bimbó 2012
6. Combinatory logic and lambda calculus are equal, algebraically (Kaposi, Altenkirch, Sinkarovs and Végh, FSCD 2023)
7. In combinator calculus
   - $\mathsf{K}uf = u$
   - $\mathsf{S}fgu = fu(gu)$

   In lambda calculus
   - $\lambda uf.u$
   - $\lambda fgu.fu(gu)$

A model of combinator calculus in Agda

```
record Model : Set₁ where
  infixl 5 _·_
  field
    Tm  : Set
    _·_ : Tm → Tm → Tm
    K S : Tm
    Kβ  : ∀{u f} → K · u · f ≡ u
    Sβ  : ∀{f g u} → S · f · g · u ≡ f · u · (g · u)
```

Trivial model

```
trivial : Model
trivial = record {
          Tm = ⊤ ;
          _·_ = λ _ _ → tt ;
          K = tt ;
          S = tt ;
          Kβ =  refl ;
          Sβ = refl
          }
```

Methodology

- **Church Encoding**: Encodes data types as functions in terms of combinators.
- **Proof Techniques**: Utilizes pigeonhole principle.

No two-element model

$$proj_i \cdot u_1 \cdot u_2 \cdot u_3 = u_i$$

- $\mathsf{proj}_1 = \mathsf{B} \cdot \mathsf{K} \cdot (\mathsf{B} \cdot \mathsf{K} \cdot \mathsf{I})$
- $\mathsf{proj}_2 = \mathsf{K} \cdot (\mathsf{B} \cdot \mathsf{K} \cdot \mathsf{I})$
- $\mathsf{proj}_3 = \mathsf{K} \cdot (\mathsf{K} \cdot \mathsf{I})$

$\mathsf{B} \cdot \mathsf{K} \cdot a \cdot b \cdot c = a \cdot b$

$\mathsf{K} \cdot a \cdot b = a$

$\mathsf{I} \cdot a = a$

No two-element model

$$proj_i \cdot u_1 \cdot u_2 \cdot u_3 = u_i$$

Using the pigeonhole principle:

- Case 1: $\text{proj}_1 = \text{proj}_2$

  $$\text{true} = \text{proj}_1 \cdot \text{true} \cdot \text{false} \cdot \text{false} = \text{proj}_2 \cdot \text{true} \cdot \text{false} \cdot \text{false} = \text{false}$$

- Case 2: $\text{proj}_1 = \text{proj}_3$

  $$\text{true} = \text{proj}_1 \cdot \text{true} \cdot \text{false} \cdot \text{false} = \text{proj}_3 \cdot \text{true} \cdot \text{false} \cdot \text{false} = \text{false}$$

- Case 3: $\text{proj}_2 = \text{proj}_3$

  $$\text{true} = \text{proj}_2 \cdot \text{false} \cdot \text{true} \cdot \text{false} = \text{proj}_3 \cdot \text{false} \cdot \text{true} \cdot \text{false} = \text{false}$$

Cases of more elements

$$proj_i \cdot u_1 \cdot u_2 \ldots \cdot u_{n+1} = u_i$$

NOP \ Ind	1	2	3	4
1	I			
2	BKI	KI		
3	BK(BKI)	K(BKI)	K(KI)	
4	BK(BK(BKI))	K(BK(BKI))	K(K(BKI))	K(K(KI))

Proof

All Done!

Conclusion

- **Main Result**: There is no finite non-trivial model of the SK combinator calculus.
- **Impact**: Enhances understanding of combinatory logic with machine-verifiable proofs.
- **Next Steps**: Explore infinite models.