神奈川大学理学部　情報科学科　馬谷研究室

静的型付き言語生成フレームワークへの型単一化機能の追加

橘川 諒一　201703675

1　はじめに

静的型付き言語とは，プログラミング言語で書かれたプログラムにおいて，変数や引数，返り値などの値について，その型がプログラムの実行よりも前にあらかじめ決められている，という性質を持つ言語である．本研究では，静的型付けされたドメイン特化言語を作るための Racket ライブラリである turnstile[2] に型単一化機能を追加する．turnstile は，関数型プログラミング言語である Racket を使用するプログラマーが型付き言語を作成できるようにすることを目的としたライブラリである．

turnstile には，手続き的型規則は定義できるが，制約（単一化）ベースの型規則を定義できない，という問題がある．そこで本研究では，turnstile に単一化機能を追加し，制約ベースの型規則を表現可能にすることを目的とする．

2　背景

ここでは，turnstile の型規則の書き方と単一化の説明を行う．

初めに，turnstile の型規則の書き方の例として，λ (ラムダ) 式の型規則を turnstile 上で表示したコードを以下に示す．

```
(define-typed-syntax (λ ([x:id : τ
_in:type] ...) e) ≫
 [[x ≫ x- : τ_in.norm] ... /- e
   ≫ e- ⇒ τ_out]
 -------
 [/- (λ- (x- ...) e-)
   ⇒ (→ τ_in.norm ... τ_out)])
```

上記の規則において，

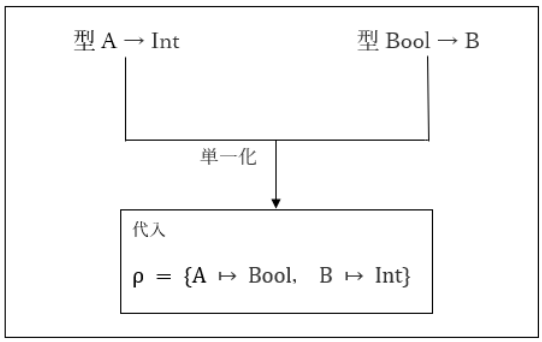

図 1　単一化と代入

```
(λ ([x:id : τ_in:type] ...) e)
```

が入力部分で，

```
[/- (λ- (x- ...) e-)
  ⇒ (→ τ_in.norm ... τ_out)]
```

が出力部分となる．

ソースコードの読み方は，λ 式は，型を持つ 1 つ以上の仮引数 x と本体式 e からなる．仮引数 x をマクロ展開したものを x-とし，τ_in.norm 型を持つことを追加する．追加した本体式 e をマクロ展開した e-は，τ_out 型を持つ．よって，マクロ展開された λ 式は，1 つ以上の仮引数が τ_in.norm 型から τ_out 型を持つ，という結論になる．

本研究の型単一化による型推論が可能になれば，仮引数のところに型を明示しなくても良くなる．

単一化の例を図 1 に示す．

単一化を説明するにあたって，型変数と代入が必要となる．型変数（図 1 での A や B がこれにあたる）とは，正体が分からない不明の型を表す．代入は，型推論の過程で明らかになる型変数の正体の対応関係のことを指す．以上より，単一化とは，（型変数を含む）型が 2 つ与えられたとき，型変数を適

当な型に置き換えて，それら 2 つを一致させることを指す．

3 設計

2 節に示した turnstile 上の λ 式を以下のように変更する．

```
(define-typed-syntax
  (λ ([x:id] ...) e) ≫
   [[x ≫ x- : τ_in] ... /- e ≫ e-
    ⇒ τ_out]
   -------
   [/- (λ- (x- ...) e-)
    ⇒ (→ τ_in ... τ_out)])

(define-typed-syntax
  (#%app e_fn e_arg ...) ≫
    [/- e_fn ≫ e_fn-
     ⇒ (~→ τ_in ... τ_out)]
    [/- e_arg ≫ e_arg-
     ⇒ τ_arg] ...
    (map unifier
     '(τ_in ...) '(τ_arg ...))
    -------
    [/- (#%app- e_fn- e_arg- ...)
     ⇒ τ_out])
```

上記のコード#%app 式に単一化する関数を呼び出すコードを追加し，λ 式の引数部分から型の入力を削除し，型変数を定義することによって turnstile で書くことができるようになる．

4 実装

ここでは，前節の規則中から呼び出されている単一化関数 unifier の実装について説明する．

```
(define unifier
 (lambda (ty1 ty2 subst exp)
  (let ...)
    (cond
     ((equal? ty1 ty2) subst)
     ((tvar? ty1)
        (if (no-occurrence? ty1 ty2)
            (extend-subst subst ty1 ty2)
            ...))
     ((tvar? ty2)
        ...)
     ((and (proc? ty1) (proc? ty2))
        (let ...))
     (else ...)))))
```

1. 4 つの仮引数を受け取り，let 式で 2 つの型に新しく代入をセットし，cond 式で評価する．
2. 2 つの型が同じなら代入を返す．
3. ty1 が型変数なら，if 式を評価する．条件は ty2 で型変数が発生しているなら extend-sub を評価し，値を返し，発生していないならエラーを返す．また，ty2 が型変数の場合は ty1 と ty2 を入れ替えて評価する．
4. 2 つの型が共に真なら let 式で 2 つの型を仮引数に変換し，代入の初期値として返す．
5. 最後に，cond 式のどれにも当てはまらなければエラーを返す．

以上を実装することにより，turnstile に型単一化の機能を追加することができる．

5 まとめ

本研究では，静的型付き言語生成フレームワークへの型単一化機能の追加を行った．turnstile ライブラリに存在しない型単一化機能の追加の為に，型単一化を定義し，実装することで turnstile の問題点を解決した．

参考文献

[1] Daniel P. Friedman, Mitchell Wand, Essentials of Programming Languages third edition, The MIT Press; third 版, 2008.

[2] Stephen Chang, Alex Knauth, and Ben Greenman. Type systems as macros. SIGPLAN Not. 52, 1 (January 2017), 694–705. 2017. DOI:https://doi.org/10.1145/3093333.3009886