

# The Definition of Snail Programming Language

letexpr

2020 年 7 月 18 日

## 1 はじめに

Snail は静的型付けの関数型プログラミング言語である.

主な特徴として,

- Bounded Linear Type によるリソースの制御
- Effect System / Coeffect System (未実装)
- 軽量の依存型 (indexed type) (未実装)

が挙げられる.

本文では Snail について Core 言語を定義し, Core 言語への脱糖規則, Core 言語の型付け規則, 操作的意味論を定義することにより Snail に定義を与える.

本文中ではメタ変数として以下のようなものを用いる.

- $\Gamma, \Delta, \Theta \dots$  型環境上を動くメタ変数.
- $A, B, C \dots$  型の上を動くメタ変数.
- $Ac, Bc, Cc \dots$  コンストラクタ上を動くメタ変数.
- $x, y, z \dots$  変数上を動くメタ変数.
- $p, q, r \dots$  resource semiring 上を動くメタ変数.
- $e \dots$  項の上を動くメタ変数.
- $s \dots$  文字列リテラル上を動くメタ変数.
- $n \dots$  自然数上を動くメタ変数.
- $i \dots$  整数上を動くメタ変数.
- $f \dots$  小数上を動くメタ変数.
- $b \dots$  論理値リテラル上を動くメタ変数.

## 2 Snail の構文定義

EBNF 記法を用いて Snail の具象構文を以下に示す.

```
toplevel ::= let [rec] x {y [ : <type>]] : <type> = <term> {<mutual-recursion-let>}  
          | typedef A = [ | ] {<Ac [of <type>]] | } <Bc [of <type>]] {<mutual-recursion-type>}
```

```
mutual-recursion-type ::= and A = [ | ] {<Ac [of <type>]] | } <Bc [of <type>]]
```

```
type ::= <type> → <type>  
       | ! '[' <expmod> ']' '{' <type> '}'  
       | <type> <type>  
       | '(' <type> ')'   
       | A
```

```
expmod ::= n | ∞
```

```
pattern ::= <pattern> <pattern>  
          | <pattern> binop <pattern>  
          | '(' <pattern> ')'   
          | x | Ac  
          | list      (組み込みリストの構文糖衣)  
          | [ ]  
          | -
```

```
mutual-recursion-let ::= and x {y [ : <type>]] : <type> = <term>
```

```
term ::= <term> <term>  
       | let [rec] x {y [ : <type>]] : <type> = <term> {<mutual-recursion-let>} in <term>  
       | fun {x [ : <type>]] → <term>  
       | match <term> with [ | ] {<pattern> → <term> | } <pattern> → <term>  
       | if <term> then <term> else <term>  
       | fix x.<term>  
       | '(' <term> [ : <type>] ')'   
       | ! <term>  
       | i | f | s | b | x | Ac | [ ] | list
```

### 3 Snail の Core 言語

Snail の Core 言語は Snail のプログラムを脱糖する事により得ることができる.

#### 3.1 Core 言語の構文

Core 言語は次のような構文を持つ.

$$\begin{aligned}
 e ::= & \text{let } !x = e_1 \text{ in } e_2 \\
 & \mid \text{i} \mid \text{f} \mid \text{s} \mid \text{b} \mid \text{x} \mid !e \\
 & \mid \text{match } e \text{ with } \{ \text{pat} \rightarrow e \mid \} \text{ pat} \rightarrow e \\
 & \mid e_1 \ e_2 \mid \lambda x. e \mid \text{fix } x. e
 \end{aligned}$$

$$\text{pat} ::= \text{pat}_1 \ \text{pat}_2 \mid \text{x}$$

#### 3.2 Core 言語の型システム

Core 言語の型付け規則を次に示す.

##### 3.2.1 Context と演算の定義

Core 言語での Context を次のように定義する.

$$\Gamma ::= \Phi \mid \Gamma, x : A \mid \Gamma, x : [A]_p$$

また, Context 間の加算  $+$  を次のように定義する.

$$\begin{aligned}
 & \Phi + \Delta = \Delta \\
 (x : [A]_p, \Gamma) + (x : [A]_q, \Delta) &= x : [A]_{p+q}, (\Gamma + \Delta) \\
 (x : [A]_p, \Gamma) + \Delta &= x : [A]_p, (\Gamma + \Delta) \quad \text{if } x \notin \Delta \\
 (x : A, \Gamma) + \Delta &= x : A, (\Gamma + \Delta) \quad \text{if } x \notin \Delta
 \end{aligned}$$

同様に, Context 間の減算  $-$  を次のように定義する.

$$\begin{aligned}
 & \Delta - \Phi = \Delta \\
 (x : [A]_p, \Gamma) - (x : [A]_q, \Delta) &= x : [A]_{p-q}, (\Gamma - \Delta) \quad \text{if } p \geq q \\
 (x : [A]_p, \Gamma) - \Delta &= x : [A]_p, (\Gamma - \Delta) \quad \text{if } x \notin \Delta \\
 (x : A, \Gamma) - \Delta &= x : A, (\Gamma - \Delta) \quad \text{if } x \notin \Delta
 \end{aligned}$$

同様に, Context と自然数の乗算  $\star$  を次のように定義する.

$$\begin{aligned} r \star \Phi &= \Phi \\ r \star (x : [A]_p, [\Gamma]) &= x : [A]_{r \star p}, r \star [\Gamma] \end{aligned}$$

$[\Gamma]$  と表記した際, Context  $\Gamma$  内には線形な変数は含まれない.

### 3.2.2 部分型付け規則

$$\begin{array}{ll} \frac{}{A <: A} & (\text{O-I}) \\ \frac{A <: B \quad q \preceq p}{!_p A <: !_q B} & (\text{O-B}) \\ \frac{A' <: A \quad B <: B'}{A \multimap B <: A' \multimap B'} & (\text{O-L}) \\ \frac{A <: B \quad q \preceq p}{[A]_p <: [B]_q} & (\text{O-D}) \\ \frac{}{\Gamma <: \Gamma} & (\text{O-IC}) \\ \frac{\Gamma <: \Delta \quad A <: B}{\Gamma, x : B <: \Delta, x : A} & (\text{O-C}) \end{array}$$

### 3.2.3 型付け規則

$$\begin{array}{ll} \frac{}{\vdash i : \text{Int}} & (\text{INT}) \\ \frac{}{\vdash f : \text{Float}} & (\text{FLOAT}) \\ \frac{}{\vdash s : \text{String}} & (\text{STRING}) \\ \frac{}{\vdash b : \text{Bool}} & (\text{BOOL}) \\ \frac{}{x : A \vdash x : A} & (\text{ID}) \\ \frac{[\Gamma] \vdash e : B}{r \star [\Gamma] \vdash !e : !_r B} & (\text{PR}) \\ \frac{\Gamma, x : A \vdash e : B}{\Gamma, x : [A]_1 \vdash e : B} & (\text{DER}) \\ \frac{\Delta \vdash e : B \quad \Gamma <: \Delta}{\Gamma, \Theta \vdash e : B} & (\text{SUB}) \\ \frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x. e : A \multimap B} & (\text{ABS}) \\ \frac{\Gamma \vdash e : A \multimap B \quad \Delta \vdash e' : A}{\Gamma + \Delta \vdash e e' : B} & (\text{APP}) \\ \frac{[\Gamma], x : [A]_p \vdash e : A \quad 1 + p \star q \preceq q}{q \star [\Gamma] \vdash \text{fix } x. e : A} & (\text{FIX}) \\ \frac{\Gamma \vdash e : !_r A \quad \Delta, x : [A]_r \vdash e' : B}{\Gamma + \Delta \vdash \text{let } !x = e \text{ in } e' : B} & (\text{LET}) \end{array}$$

### 3.2.4 アルゴリズムの型付け規則

$\frac{}{\vdash i \downarrow \text{Int}; \phi}$	(INT)	$\frac{\Delta \vdash e \downarrow B; \Gamma_2 \quad \Gamma_1 <: \Delta}{\Gamma_1, \Theta \vdash e \downarrow B; \Gamma_2}$	(SUB)
$\frac{}{\vdash f \downarrow \text{Float}; \phi}$	(FLOAT)	$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1 \vdash \lambda x. e \downarrow A \multimap B; \Gamma_2}$	(ABS)
$\frac{}{\vdash s \downarrow \text{String}; \phi}$	(STRING)		
$\frac{}{\vdash b \downarrow \text{Bool}; \phi}$	(BOOL)	$\frac{\Gamma_1 \vdash e \uparrow A \multimap B; \Gamma_2 \quad \Gamma_1 - \Gamma_2 \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash e e' \downarrow B; \Gamma_2 + \Gamma_3}$	(APP)
$\frac{}{x : A \vdash x \uparrow A; x : [A]_1}$	(ID)	$\frac{[\Gamma_1], x : [A]_p \vdash e \downarrow A; [\Gamma_2] \quad 1 + p \star q \preceq q}{q \star [\Gamma_1] \vdash \text{fix } x. e \downarrow A; q \star [\Gamma_2]}$	(FIX)
$\frac{[\Gamma_1] \vdash e \downarrow B; [\Gamma_2]}{r \star [\Gamma_1] \vdash !e \downarrow !_r B; r \star [\Gamma_2]}$	(PR)		
$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1, x : [A]_1 \vdash e \downarrow B; \Gamma_2}$	(DER)	$\frac{\Gamma_1 \vdash e \uparrow !_r A; \Gamma_2 \quad \Gamma_1 - \Gamma_2, x : [A]_r \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash \text{let } !x = e \text{ in } e' \downarrow B; \Gamma_2 + \Gamma_3}$	(LET)

## 4 参考文献