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)
- 軽量な依存型 (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.

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

$$
\begin{array}{rcl}
\text{toplevel} & ::= & \text{let [rec] x } \{\text{y [: } \langle\text{type}\rangle]\}\text{: } \langle\text{type}\rangle = \langle\text{term}\rangle\ \{\langle\text{mutual-recursion-let}\rangle\} \\
 & | & \text{typedef A } = [\ |\ ]\ \{\langle\text{Ac [of } \langle\text{type}\rangle]\rangle\ |\ \}\ \langle\text{Bc [of } \langle\text{type}\rangle]\rangle\ \{\langle\text{mutual-recursion-type}\rangle\} \\
 & & \\
\text{mutual-recursion-type} & ::= & \text{and A } = [\ |\ ]\ \{\langle\text{Ac [of } \langle\text{type}\rangle]\rangle\ |\ \}\ \langle\text{Bc [of } \langle\text{type}\rangle]\rangle \\
 & & \\
\text{type} & ::= & \langle\text{type}\rangle \to \langle\text{type}\rangle \\
 & | & !\ \text{'['}\ \langle\text{expmod}\rangle\ \text{']'}\ \text{'\{'}\ \langle\text{type}\rangle\ \text{'\}'} \\
 & | & \langle\text{type}\rangle\ \langle\text{type}\rangle \\
 & | & \text{'('}\ \langle\text{type}\rangle\ \text{')'} \\
 & | & \text{A} \\
 & & \\
\text{expmod} & ::= & \text{n}\ |\ \infty \\
 & & \\
\text{pattern} & ::= & \langle\text{pattern}\rangle\ \langle\text{pattern}\rangle \\
 & | & \langle\text{pattern}\rangle\ \text{binop}\ \langle\text{pattern}\rangle \\
 & | & \text{'('}\ \langle\text{pattern}\rangle\ \text{')'} \\
 & | & \text{x}\ |\ \text{Ac} \\
 & | & \mathit{list} \qquad \text{(組み込みリストの構文糖衣)} \\
 & | & [\,] \\
 & | & \_ \\
 & & \\
\text{mutual-recursion-let} & ::= & \text{and x } \{\text{y [: } \langle\text{type}\rangle]\}\text{ : } \langle\text{type}\rangle = \langle\text{term}\rangle \\
 & & \\
\text{term} & ::= & \langle\text{term}\rangle\ \langle\text{term}\rangle \\
 & | & \text{let [rec] x } \{\text{y [: } \langle\text{type}\rangle]\}\text{: } \langle\text{type}\rangle = \langle\text{term}\rangle\ \{\langle\text{mutual-recursion-let}\rangle\}\ \text{in}\ \langle\text{term}\rangle \\
 & | & \text{fun } \{\text{x [: } \langle\text{type}\rangle]\}\ \to \langle\text{term}\rangle \\
 & | & \text{match}\ \langle\text{term}\rangle\ \text{with}\ [\ |\ ]\ \{\langle\text{pattern}\rangle \to \langle\text{term}\rangle\ |\ \}\ \langle\text{pattern}\rangle \to \langle\text{term}\rangle \\
 & | & \text{if}\ \langle\text{term}\rangle\ \text{then}\ \langle\text{term}\rangle\ \text{else}\ \langle\text{term}\rangle \\
 & | & \text{fix x.}\langle\text{term}\rangle \\
 & | & \text{'('}\ \langle\text{term}\rangle\ [\text{ : }\langle\text{type}\rangle]\ \text{')'} \\
 & | & !\ \langle\text{term}\rangle \\
 & | & \text{i}\ |\ \text{f}\ |\ \text{s}\ |\ \text{b}\ |\ \text{x}\ |\ \text{Ac}\ |\ [\,]\ |\ \text{list}
\end{array}
$$

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.

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

$$
\mathrm{pat} ::= pat_1\ pat_2 \mid \mathrm{x}
$$

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.

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

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.

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

3.2.2　部分型付け規則

$$\frac{}{A <: A} \quad \text{(O-I)}$$

$$\frac{A <: B \quad q \preceq p}{[A]_p <: [B]_q} \quad \text{(O-D)}$$

$$\frac{A <: B \quad q \preceq p}{!_p A <: !_q B} \quad \text{(O-B)}$$

$$\frac{}{\Gamma <: \Gamma} \quad \text{(O-IC)}$$

$$\frac{A' <: A \quad B <: B'}{A \multimap B <: A' \multimap B'} \quad \text{(O-L)}$$

$$\frac{\Gamma <: \Delta \quad A <: B}{\Gamma, x : B <: \Delta, x : A} \quad \text{(O-C)}$$

3.2.3　型付け規則

$$\frac{}{\vdash i : \mathrm{Int}} \quad \text{(INT)}$$

$$\frac{\Delta \vdash e : B \quad \Gamma <: \Delta}{\Gamma, \Theta \vdash e : B} \quad \text{(SUB)}$$

$$\frac{}{\vdash f : \mathrm{Float}} \quad \text{(FLOAT)}$$

$$\frac{}{\vdash s : \mathrm{String}} \quad \text{(STRING)}$$

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma \vdash \lambda x.e : A \multimap B} \quad \text{(ABS)}$$

$$\frac{}{\vdash b : \mathrm{Bool}} \quad \text{(BOOL)}$$

$$\frac{\Gamma \vdash e : A \multimap B \quad \Delta \vdash e' : A}{\Gamma + \Delta \vdash e\ e' : B} \quad \text{(APP)}$$

$$\frac{}{x : A \vdash x : A} \quad \text{(ID)}$$

$$\frac{[\Gamma], x : [A]_p \vdash e : A \quad 1 + p \star q \preceq q}{q \star [\Gamma] \vdash \mathrm{fix}\ x.e : A} \quad \text{(FIX)}$$

$$\frac{[\Gamma] \vdash e : B}{r \star [\Gamma] \vdash !e : !_r B} \quad \text{(PR)}$$

$$\frac{\Gamma, x : A \vdash e : B}{\Gamma, x : [A]_1 \vdash e : B} \quad \text{(DER)}$$

$$\frac{\Gamma \vdash e : !_r A \quad \Delta, x : [A]_r \vdash e' : B}{\Gamma + \Delta \vdash \mathrm{let}\ !x = e\ \mathrm{in}\ e' : B} \quad \text{(LET)}$$

3.2.4　アルゴリズム的型付け規則

$$\frac{}{\vdash i \downarrow \text{Int}; \phi} \quad (\textsc{int})$$

$$\frac{}{\vdash f \downarrow \text{Float}; \phi} \quad (\textsc{float})$$

$$\frac{}{\vdash s \downarrow \text{String}; \phi} \quad (\textsc{string})$$

$$\frac{}{\vdash b \downarrow \text{Bool}; \phi} \quad (\textsc{bool})$$

$$\frac{}{x : A \vdash x \uparrow A; x : [A]_1} \quad (\textsc{id})$$

$$\frac{[\Gamma_1] \vdash e \downarrow B; [\Gamma_2]}{r \star [\Gamma_1] \vdash\ !e \downarrow !_r B; r \star [\Gamma_2]} \quad (\textsc{pr})$$

$$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1, x : [A]_1 \vdash e \downarrow B; \Gamma_2} \quad (\textsc{der})$$

$$\frac{\Delta \vdash e \downarrow B; \Gamma_2 \qquad \Gamma_1 <: \Delta}{\Gamma_1, \Theta \vdash e \downarrow B; \Gamma_2} \quad (\textsc{sub})$$

$$\frac{\Gamma_1, x : A \vdash e \downarrow B; \Gamma_2}{\Gamma_1 \vdash \lambda x.e \downarrow A \multimap B; \Gamma_2} \quad (\textsc{abs})$$

$$\frac{\Gamma_1 \vdash e \uparrow A \multimap B; \Gamma_2 \qquad \Gamma_1 - \Gamma_2 \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash e\ e' \downarrow B; \Gamma_2 + \Gamma_3} \quad (\textsc{app})$$

$$\frac{[\Gamma_1], x : [A]_p \vdash e \downarrow A; [\Gamma_2] \qquad 1 + p \star q \preceq q}{q \star [\Gamma_1] \vdash \text{fix } x.e \downarrow A; q \star [\Gamma_2]} \quad (\textsc{fix})$$

$$\frac{\Gamma_1 \vdash e \uparrow !_r A; \Gamma_2 \qquad \Gamma_1 - \Gamma_2, x : [A]_r \vdash e' \downarrow B; \Gamma_3}{\Gamma_1 \vdash \text{let } !x = e \text{ in } e' \downarrow B; \Gamma_2 + \Gamma_3} \quad (\textsc{let})$$

4　参考文献